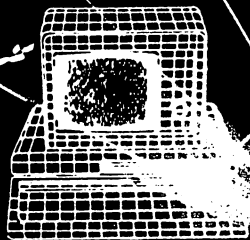
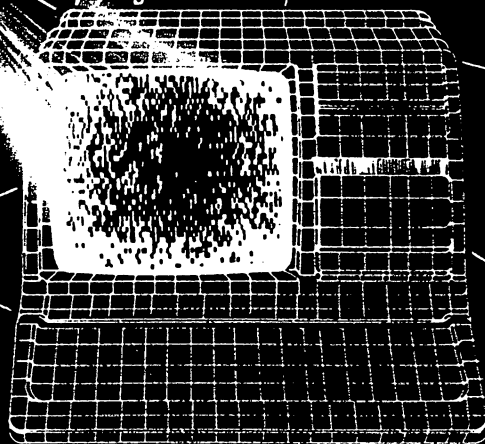


THE ULTIMATE IN COMMUNICATIONS SOFTWARE



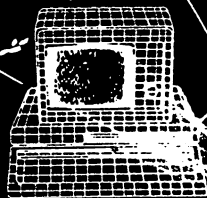
OnLine!



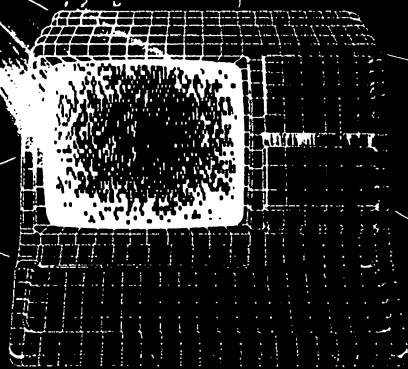
MICRO-SYSTEMS SOFTWARE, INC.



THE ULTIMATE IN COMMUNICATIONS SOFTWARE



OnLine!



MICRO-SYSTEMS SOFTWARE, INC.



Copyright

OnLine! is copyrighted (c) (p) 1985,86 by Micro-Systems Software, Inc. The OnLine! Users Manual is copyrighted (c) (p) 1985,86 by Micro-Systems Software, Inc. All rights are reserved. You may not make any copies of the documentation, electronic or otherwise, without the express written permission of the publisher.

Warranty

Micro-Systems Software warrants that this program will perform as documented in this manual, and warrants that the media will be free from defects. This warranty is limited, and Micro-Systems Software specifically disclaims any implied warranties of merchantability or fitness for a particular purpose. Micro-Systems Software's warranty is limited to repair or replacement of the defective media. Micro-Systems Software shall not be liable for any damages, whether consequential or incidental, that arise from the use or misuse of this product. In NO event shall the total amount of Micro-Systems Software's liability exceed the purchase price of the product. No returns accepted after 30 days. All returns may be subject to a re-stocking fee.

Backups and Duplicates

Micro-Systems Software grants to the purchaser of this product the right to make duplicates for PERSONAL BACKUP ONLY. The purchaser does not have the right to distribute these duplicates, regardless of whether or not a fee is charged for such distribution. This program is not copy-protected, and may be installed on a hard disk at will. However, installation on multiple CPUs is specifically prohibited. Reasonable corporate site licenses are available.

Technical Support

If you have questions about, or problems with, OnLine!, you can write or call:

Micro-Systems Tech Support
4301-18 Oak Circle
Boca Raton, FL 33431
(305) 391-5077

Technicians are available to answer your questions from 9 a.m. until 5 p.m. Monday through Friday, Eastern Standard Time. Less pressing matters can be referred to the BBS.

Bulletin Board Support

Micro-Systems Software maintains a bulletin board online 24 hours a day, 7 days a week. You may call here to ask questions during non-business hours. The bulletin board supports 300, 1200, or 2400 bits per second and any parity/word length setting. The number is:

(305) 737-1590

CompuServe Support

Visit us on the CompuServe SIG we SYSOP; the TeleComm Group. Type GO TELECOMM from any prompt and leave a message to SYSOP in section 4 (MSS support). Our PPN is 76703,447, if you want to use EasyPlex.

Addendum for **OnLine!** User's Manual

The following documents the changes and enhancements that have been made to **OnLine!**

- * When capturing a file to disk, we now specifically check to make sure the file does not exist before opening it. This prevents repeated captures to the same file from truncating prior data.
- * The Archive requester now remains on the screen until Resume or Get is selected. Also, when using Replace, the comment portion of the filename is no longer lost.
- * The Display_Beep() routine is no longer used. **OnLine!** actuates the sound device and generates REAL noise!
- * VT-100 emulation was not removing attributes (boldface, reverse video, etc.) from text. This has been repaired.
- * The Parallel device is now opened in "shared" mode.
- * When a character is translated into a "null" value (00), it will no longer be sent to any of the system devices.

Two major additional functions have been added. First, CompuServe's "B" protocol is now supported for upload and download. It operates in two modes. First, when using it with CompuServe itself, select "B" protocol on the Transfer Protocol menu. **OnLine!** will respond automatically to commands sent from CompuServe. You do not need to tell **OnLine!** whether to receive or send. CompuServe will do that, even down to the filename. When using the "B" protocol between two copies of **OnLine!**, treat it exactly as you would any of the other protocols. Select "B" protocol and then either Receive or Send.

For example, when you want to download a file from CompuServe, you begin by selecting "CIS-B" on the Transfer

Protocol sub-menu. This can be done before or after you connect with CompuServe, but it must be done before you begin the file transfer sequence. You enter the download command to CompuServe and are asked which protocol to use. Select "B" protocol from the menu presented to you. CompuServe will ask for a "Filename for your computer." Enter the filename as YOU want to store it locally (e.g., "DF1:COLOR-GAME"). The file transfer window will automatically appear when CompuServe initiates the file transfer. Uploading a file is the exact reverse of this. The important point here is that you tell CompuServe the filename and it echoes that back to OnLine!. You do not actually select Send or Receive when using "B" protocol transfers with CompuServe. You do, however, when using the protocol between copies of OnLine!.

Second, when programming strings (as in macrokeys or script files), you can now enter a percent sign (%) followed by a decimal ASCII value for special characters. The next time the character is displayed, the character itself will appear, following the percent sign, not the decimal value. This was included so that characters used internally by OnLine!, such as the vertical bar character for a carriage return, could now be used in "wait strings" and macrokeys. No special ending character is needed when using this function. The input stops at the first non-numeric character.

A final note. When using OnLine!'s VT-100 emulation, the following special key assignments are in effect. The PF1 through PF4 keys are emulated with SHIFT-F1 through SHIFT-F4 on the Amiga keyboard. The plus and minus keys on the numeric keypad function as their VT-100 counterparts, and the HELP key is the keypad comma.

OnLine! Version 1.0

Table of Contents

Getting Started	1-1
Introduction	1-1
Overview	1-4
What You Need to Use OnLine!	1-5
First Things First, Backing Up Disks	1-7
What to Do? (Accessing the Information Service)	1-8
Use OnLine! Manually First	1-10
Starting OnLine!	1-10
Using the Mouse with OnLine!	1-11
Terminal (Configuration) Files	1-12
Using an Existing Terminal File	1-13
Creating a New Terminal File	1-14
Power-up Default Values	1-15
Operating the OnLine! Keyboard	1-16
Using OnLine!'s Requestors	1-19
Setting the Phone Number	1-23
Setting Communications Parameters	1-24
Check Your Work	1-25
Saving Settings in a File	1-25
Some Common OnLine! Customizations	1-26
Conncting with the Service	1-27
Error! Number Dialed but no Contact	1-28
Error! Contact Made but Garbled	1-28
Printing a Transcript of your Session	1-29
Capturing Text from a Service	1-29
Sending a Text File to a Service	1-32
Protocol File Transfers	1-34
Conclusion	1-35

Automating the Process	2-1
What is a Script File?	2-3
Controlling Scripts	2-5
Creating an Auto-Logon Script	2-6
An Explanation of the Status Window	2-9
Conclusion	2-15
 Reference	 3-1
Introduction	3-1
Command Descriptions	3-1
Using OnLine!'s Requestors	3-2
OnLine! Command Menus	3-6
<i>Service Menu</i>	3-6
Archive	3-7
Call	3-7
Hangup	3-8
Quit	3-8
<i>Transfer Menu</i>	3-8
Receive	3-9
Send	3-9
Protocol	3-10
Xmodem	3-10
CRC - Xmodem	3-10
HVP	3-10
<i>Capture Menu</i>	3-10
Archive	3-11
Open	3-11
Close	3-13
Clear	3-13
Send	3-13
Pacing Data Transmission	3-14

Stop	3-14
Go	3-14
C-delay	3-14
L-delay	3-14
Prompt	3-15
View	3-15
Display	3-15
Printer	3-15
<i>Script Menu</i>	<i>3-15</i>
Archive	3-16
Stop	3-16
Go	3-16
Resume	3-16
Clear	3-17
View	3-17
<i>COM Menu</i>	<i>3-17</i>
Reset	3-17
Baud	3-17
Word	3-18
Parity	3-18
Stops	3-18
<i>Other Menus</i>	<i>3-19</i>
Printer	3-19
Terminal	3-19
Width	3-20
Duplex	3-21
Echo	3-21
CR+LF	3-22

Flow	3-23
Window	3-23
<i>Phone Menu</i>	3-24
Primary	3-25
Alternate	3-25
Retries	3-25
Prefix	3-26
Suffix	3-27
<i>Tables Menu</i>	3-27
Macrokey	3-27
Display	3-30
Printer	3-30
Keyboard	3-30
From - capture	3-30
To - capture	3-31
In - COM	3-31
Out - COM	3-31
Conclusion	3-31
Appendix A / Script Commands	A-1
Introduction	A-1
What is a Script File	A-3
Creating a Script File	A-3
Script File Commands	A-4
Abort	A-6
Alarm	A-6
Ask	A-6
Bye	A-7
Capture	A-8

Clear	A-9
Color	A-9
Do	A-10
If	A-11
Jump	A-12
Label	A-13
Message	A-13
Offline	A-14
Printer	A-14
Reply	A-15
Rwind	A-15
Sbreak	A-16
Skip	A-16
Wait	A-17
When	A-21
General Programming Tips	A-22
Using the Learn Mode	A-23
Changing the Script Files	A-24

Chapter One

Getting Started

Introduction

Congratulations on your selection of OnLine!. This advanced program offers many powerful and unique telecommunications features. You can use OnLine! to communicate with mainframes, local bulletin boards, or information services (such as CompuServe and The Source). OnLine! can be used as a simple terminal, or it can be used to download information from the host system and store this information on your computer. Using OnLine!'s script language, you can even have the program interact with the remote computer automatically.

The following list details some of OnLine!'s features. Each one of these will be explained in greater detail later in the manual. They are listed now to provide you with an overview of the program's capability.

- * Simple pull-down menu interface.
- * All program settings, including auto-logon scripts, may be stored in special terminal definition (.TRM) files for easy recall. You may store files for as many services as disk space permits.
- * All file functions support DOS pathnames and hard disks.

- * ASCII capture buffer for storing incoming data. Buffer space is user-definable (64K default), with an optional overflow to a disk file.
- * Captured data may be stored to disk, scrolled to the display, or printed on the lineprinter.
- * Data may be transmitted from the capture buffer to the host in either prompted mode (line by line), block mode (XON/XOFF control), or with user-defined delays between characters and/or lines.
- * Support for 300 through 19200 baud. Any word length or parity supported.
- * User-definable macrokeys can be used to transmit often used commands. Edit window makes the definition of these keys simple and fast. Macrokeys may even be transmitted under script file control.
- * Program can operate in full or half duplex. OnLine! can even echo received characters back to a calling computer for "chatting" with full duplex only terminals.
- * Text reformatting allows you to display text in a pleasing format regardless of incoming width.
- * Auto-dial support for any command driven modem. Separate user definable dial prefix and suffix. Default setting support Hayes and Hayes compatible modems.
- * Allows two phone numbers for each service, with user-definable retries (up to 99). If connection is not made on the primary number, the alternate will be tried.

- * 7 separate translation tables permit you to change any one byte value into any other value for all system devices.
- * User can control window size (79x22 with frame, 80x23 without). ANSI escape sequences supported in TTY emulation mode.
- * Special "chat" window provides local echo regardless of incoming data. Excellent for CompuServe's CB Simulator.
- * Hardcopy support. OnLine! can send received characters to both the screen and the printer.
- * OnLine! can use script files for automated operation (not just auto-logon, but entire automated sessions -- even downloading). Script files can prompt for operator input and act on the result. An entire programming language with OnLine!!
- * Script files can be created online, in a simple question and answer style, with OnLine!'s unique "learn mode".
- * Direct to disk file transfer is supported in four different protocols: XMODEM (Christensen), XMODEM/CRC, Hayes Verification Protocol (compatible with SmartCom), and standard text capture to disk. XMODEM protocol is relaxed for use with services such as CompuServe, especially over TymeNet.

As you can see from the above list, OnLine! is a sophisticated program. In order to help you make best use of the program, this manual is organized into two distinct parts. The first part is the OnLine! Tutorial. This portion of the manual is designed to have you using your new program to call another computer just as soon as possible. We'll also discuss certain major aspects of using the program, such as capturing data and printing a hardcopy of your communications session.

This will be accomplished in a general overview of the program and a specific discussion regarding the use of script files in automating OnLine!.

The second part of the manual is the OnLine! Reference Manual. Here we begin covering the command menus one option at a time. This approach is taken because once you begin using the program, your questions are usually specific in nature. You will find this reference manual a valuable aid when you want to know "just exactly what does the Open option on the Capture menu do?".

If you are a seasoned telecommunications and Amiga veteran already and just want to get rolling, boot up your Amiga using Kickstart 1.1 and the supplied OnLine! diskette and run the program from WorkBench. Once the program is running, press the Help key to get the general idea how things work. Begin to try out the various menus. You can't really hurt anything. However, if you have the time, the tutorial should still make interesting reading.

Overview

The telecommunications field is one of the fastest growing of all the many areas where microcomputers are involved. It may be true that soon no-one will be able to order groceries without a terminal in their home. But even in a more reasonable scenario, computer users will be increasing in their daily dependence upon modems and outside databases.

There are two usual functions for someone using a computer and modem. First would be calling a local bulletin board system, and second would be calling one of the many "information networks" (perhaps the largest of which is CompuServe, a division of H & R Block, Inc.). The business professional making use of telecommunications will have additional functions such as connecting with the mainframe computer at their company, transmitting data and reports from

branch offices to the home office, and perhaps even working from home (a concept known as “telecommuting”, which refers to “traveling” to one’s job via modem).

OnLine! will meet the requirements of all these applications easily. Whether you are interested in the latest stock quote from the Dow Jones News Retrieval Service or downloading free programs from the local bulletin board systems, you’ll find a wealth of commands in OnLine! that make this task simpler.

There is a special thrill in realizing that you can be connected via ordinary telephone lines to a distant computer like your own or a large mainframe computer. You can even be involved in a data conference where many computers across the country may be interconnected via a central network.

What You Need to use OnLine!

First of all, you need an Amiga with 256K of memory and one disk drive. (OnLine! will also work with additional disk drives and memory, should you have those.) OnLine! stays very carefully within the conventions of AmigaDOS, so future enhancements in the area of memory or hard disks should not bother OnLine! at all.

In addition to OnLine! and your computer, you will also need a MODEM (short for “modulator/demodulator”). This device will permit your computer to send and receive data across telephone lines. Modems are divided into two types, “acoustic couplers” and “direct connect”.

The acoustic coupler modem will attach in some manner to your telephone handset. These require you to dial the remote system, listen for the tone, and place the handset into

the modem. Acoustic couplers are very noise sensitive and not usually available in speeds faster than 300 bps.

A direct connect modem, on the other hand, attaches directly to your phone wall jack and the serial port of your computer. These tend to provide a more reliable connection and are available in faster speeds (1200 and 2400 bps).

Obviously, acoustic coupler modems will not use the auto-dial capabilities of OnLine! (since you have to dial the phone). OnLine! will still function fine with such a modem, since OnLine! does not INSIST on dialing the modem in order to communicate. Most direct connect modems have auto-dial capability. This means they can dial the remote computer, under instruction from OnLine!.

The majority of auto-dial, direct connect modems accept a command from the computer over the serial port as an instruction to dial. This is called a "command driven" modem. (Other direct connect modems require that a logic signal, such as RTS, is "pulsed" in order to dial the phone. OnLine! will NOT auto-dial such a modem.) OnLine! will work with any command driven modem, allowing you to configure the dialing "prefix" (what is sent BEFORE the phone number), the phone number, and the dialing "suffix" (what is sent AFTER the phone number). OnLine! is already set up for the Hayes Smartmodem (or a compatible unit), since Hayes is by far the most common brand of modem and many other brands still use the Hayes command set for dialing.

IMPORTANT NOTE: You will also need a special modem cable to be made for you. The Amiga's serial port contains several signals that could be damaging to your modem without a cable that is "missing" some pins. Do NOT use a serial cable that has all 25 pins connected straight through! This same situation also exists with the parallel printer cable. The only pins that should be present in the cable end are:

1	Frame Ground
2	Transmit Data
3	Receive Data
4	Request to Send
5	Clear to Send
6	Data Set Ready
7	Signal Ground
8	Data Carrier Detect
12	High Speed Signal
20	Data Terminal Ready
22	Ring Indicator

You must either purchase a custom cable or an "off the shelf" cable DESIGNED to work with the Amiga. If your dealer cannot help you with this, Micro-Systems can sell you an Amiga modem cable.

First Things First, Backup the Diskette!

Before we get too deeply into running the program, the first item you should concern yourself with is making a duplicate copy of the master diskette. You may backup these diskettes with DISKCOPY. If you have questions about how to operate the DISKCOPY utility, please consult the user's manual that came with your copy of DOS.

OnLine! is delivered on a copy of AmigaDOS/WorkBench (under license from Commodore-Amiga, Inc.), and so the backup copy of your master OnLine! diskette may be inserted in the drive when the Amiga requests WorkBench.

IMPORTANT NOTE: OnLine! was written to be compatible with Kickstart 1.1, and requires that version to operate. You cannot use it with an earlier version of Kickstart. If an earlier version of Kickstart is loaded into your Amiga, you will need to turn the computer on and off to reload Kickstart, making sure that you use version 1.1.

Once you have duplicated the master diskette, file it away and do not use it again, save to make additional copies. You should use the duplicate copy as your working disk. Never use the actual OnLine! master diskette. This is NOT copy protected, so there is no reason for continuing to use the program from the master diskette and later suffering a disk failure.

What to Do? (Connecting up)

OK, so you've backed up the OnLine! diskette and you're ready to get started. Well, before you can learn HOW to call, you must decide WHO to call.

There are a number of commercial and special interest information services that you can contact. Information services usually charge by the hour; few have toll-free numbers (although there are low-cost telephone networks that substantially reduce the cost of long distance telecommunications); some use special "dial-up" networks that offer connection via a local phone number for an additional surcharge. Charges for commercial services may run from \$5.00 per hour to over \$35.00 per hour (many times depending upon the time of day you access the service).

Some commercial services offer securities and trading data, along with an up-to-the minute news wire complete with dining guides, movie reviews and other features.

However, most of these information services require that you buy an access package in advance from your local computer dealer. A few will provide online signup options, where first time callers that wish to register a credit card number can join up at once. The majority will not, though. So if your desire is to dial up CompuServe or The Source, you should be certain that you have the proper user I.D. codes and passwords to access them before you call them up.

There are also many local bulletin boards designed for users of specific computers, individuals with common interests from agriculture to photography, even members of certain professions. Local computer retailers frequently sponsor electronic bulletin boards as a service to customers. Normally, these bulletin boards are free (except for long distance charges that may apply), although there is often a registration process you must go through before you get complete access. These bulletin boards are often an excellent source of public domain software that you may use to enhance your computing experience.

Another interesting application is that it is possible to connect directly with Western Union and the international Telex network. Your personal computer can also become an electronic mail terminal with services such as MCI Mail which allow you to send mailgrams, executive letters and messages automatically from your office to anyone, virtually anywhere in the world.

Where do you get a list of these phone numbers? If you choose one of the commercial information services, the package you get when you purchase your access codes will usually contain the access numbers and instructions. But what of the local bulletin boards?

Many computer magazines will regularly print such lists and local bulletin boards (once you find the first one) will often carry a list of other such systems in their area or even nationwide. (There are no guarantees with such a list, though. The speed at which new bulletin boards come and go is staggering.)

Perhaps your computer dealer operates a bulletin board at their store? Many do. This is a good place to begin. If all else fails, you may call Micro-Systems' bulletin board listed earlier. This WILL be long distance for most of you, though. Be prepared.

Use OnLine! Manually First

Although OnLine!'s automation is exceptional, you should first learn how to operate the various basic elements of connecting up with someone else. It will be easier to tell you HOW to automate when you know WHAT you are automating. Later, when you become proficient at what the various menu items do, then you're ready to construct some comprehensive script files to automate your tasks (beyond just logging you into a service). Creating a script file to do automatic logon is much simpler, especially using the learn mode, then developing specialized scripts that navigate within a service.

A script file, by the way, is pretty much what it sounds like. In much the same way as actors follow a script where each one speaks in turn, taking their cues from what the other actors say or do, your OnLine! can follow a script file that you create. This script file will tell OnLine! what prompts to expect from the host system and what replies to make. But scripts do not stop there. They can also control the capture buffer, the printer, and other areas, all without you having to operate the keyboard.

So we'll look at those items you must configure in order to get connected up. The ABCs of communication, if you will.

Starting OnLine!

Before you can use OnLine!, you must start the program. To do this, from WorkBench, select the OnLine! icon and click the left mouse button twice. If you would rather use CLI for this task, you may log into the root directory of the program diskette and type:

OnLine!

Press the RETURN key. In EITHER case, the OnLine! program will load and the initial screen will be displayed. This is the "terminal mode". In this mode, anything you type will be transmitted out the serial port and any characters received over the serial port will be displayed on the screen. At the top of the screen is displayed a status bar with the following indicators (which will, of course, be filled in with data):

Terminal: Printer: Capture: Script:

(NOTE: That was an IMPORTANT term just mentioned, so you should take care to remember it. The "terminal mode" is where you will spend 99% of your time with OnLine!, unless you have a very specialized need for this program. It is in this mode of the program that you are interacting with the remote computer. Much mention will be made of the terminal mode throughout this manual and it will be good for you to remember what it is.)

Using the Mouse with OnLine!

OnLine! uses the normal Amiga mouse and menu system. Press the right mouse button to display the list of available pull down menus at the top of the screen. Select the pull down menu you desire by moving the mouse pointer over the menu name. Each pull down menu contains several options. To select an option, while CONTINUING to hold down the right mouse button, position the mouse pointer over the option. If another sub-menu appears, you may select an option on this menu in the same manner.

Once you have highlighted the option you wish to engage, all you need to do is release the right mouse button. If you wish to select an option and remain in the menus (as in the case of communications parameters, where you might want to select a baud rate and then a word length, and then a parity

setting, all without pausing), press the LEFT mouse button while CONTINUING to hold the right mouse button. Always select the last option by releasing the right mouse button.

When you release the right mouse button, the menus will disappear and whichever selection was highlighted at the moment will be selected. Use caution not to release the mouse button while there is an option selected that you do not wish to execute. OnLine! will not perform most permanent or terminating functions without prompting for additional verification (this would include items such as deleting files or exiting the OnLine! program).

If you select a menu option that requires additional input from you, such as a filename, OnLine! will bring up a requester asking for the information. Certain special items, such as the macrokeys and translation tables, use a very different sort of requester that is described in detail later on in this tutorial.

Terminal (configuration) Files

OnLine! uses special configuration files that are called "terminal" files and are denoted by the characters ".trm" at the end of the filename. These terminal files contain everything OnLine! needs to know about a particular service. Important information such as the phone number (for auto-dial modems), the baud rate and serial parameters (word length, parity and stop bits), how you want your display configured (whether or not to use a window border, terminal emulation, translation tables, etc.), the desired set of user programmed macrokeys, and a link to any required auto logon scripts.

The default terminal file, OnLine!.trm, will be loaded automatically, if it is present in the current logged directory at the time the program loads. This means you can create a "default" environment of your most commonly used settings

and macrokeys, and save this into a terminal file named "OnLine!" (NOTE: The upper/lower case in the name is for style's sake only, and doesn't matter to AmigaDOS). Then each time you want to make another another terminal file for a different service, all you need to do is boot up the program (which loads the default), enter a phone number, build an auto logon script, and save a terminal file with the new name.

Before you can call someone with OnLine!, you should understand the various elements that go into a terminal file. These are the "ABCs of communication" alluded to earlier in the manual.

Using an Existing Terminal (.trm) File

If you start OnLine! from CLI, as opposed to WorkBench, you may specify the name of a terminal file after the program name. For example, consider the following:

OnLine! BBS-PC

This would load OnLine! and the terminal file "BBS-PC.trm" from the currently logged directory.

If you start OnLine! from WorkBench, and want to use a terminal file other than OnLine!.trm, you will need to use the Archive option of the Service menu. (See "Using OnLine!'s requesters" below.)

If no terminal files exist, you can create one. All that is involved is activating OnLine!, setting the proper communication settings and then activating Service Archive to "store" them. Archive will automatically place ".trm" at the end of the filename you choose. What are the "proper communications settings"? Let's look at that now.

Creating a New Terminal (.trm) File

Telecommunications is a flexible medium. It is this fact that enables you to tie into an enormous number of computer networks and share data with hundreds of thousands of computers all over the world, even though they are widely differing brands and types. That kind of flexibility is the result of just a few "standards" that must be "matched" between communicating computers. The critical parameters are: baud rate, word size, parity check and stop bits.

NOTE: Some bulletin board systems will require you to press RETURN one or more times prior to logging in. From this, they will be able to determine your word length and parity settings. In these cases, all you need to match is the baud rate. Other bulletin boards and almost all commercial services will list serial parameters that you must adjust to in order to connect up.

As a minimum, before you can create a terminal file, you will need to know the following information about the "Host" computer you are calling:

Phone Number : _____

Baud Rate : _____

Word Size : _____

Parity Check : _____

Stop Bits : _____

When calling local bulletin boards, it is also best to know any password requirements and the system's hours of operation. Many bulletin boards will operate on a phone line used for voice communications at other hours. It is very bad

form to call when the system is not online, so exercise a little forethought.

Another item, password requirements, can save you a lot of aggravation. Some bulletin boards do not allow access to the general public. Access is restricted to a specific group of people, or at least limited to those callers approved by the SYSOP. If you know that a system requires a password before you can access it, either arrange for the password or don't bother to call. (It should be noted that more than a few systems allow a first time caller to register online for a password and subsequent calls will yield complete access. In these cases, you obviously must call the system even though you will not be able to properly use it.)

Power-up Default Values

OnLine! has "power up defaults" built into the software. These are used when it does not find the file OnLine!.trm in the currently logged directory. (If the file OnLine!.trm is located in the currently logged directory, whatever settings are specified by that file will be used.) When OnLine!.trm is NOT found, your settings will be:

Phone Number : unknown
Baud Rate : 1200
Word Size : 7
Parity Check : Even
Stop Bits : 1

However, each can be easily changed and stored in a terminal file. Therefore, each terminal file has its own set of defaults that may differ from the power up defaults. These can be viewed using the Examine option of the Service menu or typing Amiga-key E when you are in the terminal mode.

When a terminal file is activated, its default settings override the power up defaults, making it necessary to set

communication parameters only once for each information service or host with which you communicate.

Right now, it is only necessary to set those parameters listed above to make contact with the information service of your choice. We'll take a quick look at each of these in turn, but first we want to discuss using OnLine!'s keyboard.

Operating the OnLine! Keyboard

To make the best use of OnLine!, you should become comfortable with issuing commands to OnLine!. This is done two ways. The mouse and the keyboard.

Most of your command input to OnLine! will be done with the mouse. OnLine!'s pull-down menus contain all of the options you'll need to control the program. You do not have to remember difficult sequences of key assignments. However, certain actions will make use of the keyboard. For example, the ten function keys along the top of the keyboard are used to transmit the keyboard macros. In addition, certain of the most commonly used functions (open/close capture buffer, turn the printer echo on/off, initiate a protocol file send/receive, and others) are on "Amiga-keys".

The Amiga-keys are those bearing the stylized letter A. The right hand Amiga-key is normally the one used in command selection, supposedly to remind you of the right hand mouse button. Therefore, you will be able to hold down the right Amiga-key and select some common functions by typing a single letter on the keyboard. Which functions are on Amiga-keys can be seen by scrolling through the pull-down menus. The options with Amiga-key activation list the proper keystroke to the right of the menu option.

There are also certain commands activated by using the ALT keys (as opposed to the Amiga keys). In these cases,

hold down the ALT key instead of the Amiga key and press the letter of your choice.

We've already discussed using the mouse to select menu options, so we won't waste space going over that again.

Go ahead and spend some time examining the various menus. The best way to become familiar with a program is to "play" with it. As long as there is no option highlighted when you release the mouse button, nothing will be selected. You are free to move the mouse about at will. The menus are in logical groupings. That is, the main menu divides into general "areas" of control, while the pull-down menus that come from each command have the actual options that control this area. For example, press the right mouse button and you see the main menu, which is:

Service Transfer Capture Script COM Other Phone Tables

You decide that you want to display the information that you have in the capture buffer from your last online session. You select the Capture menu by moving the mouse pointer over the word "Capture". The pull-down menu is displayed, this one with specific capture options on it. It appears as follows:

Archive
Clear
Open
Close
Send
View

Since you want to display the information stored in the capture buffer, you select View by moving the mouse pointer down the menu until the word "View" is highlighted. A submenu appears, containing the choices:

Display
Print

Now you must choose between displaying these contents on your monitor or your line printer. Deciding that the

monitor alone should suffice (the Print option displays data on BOTH the monitor and the line printer), you move the mouse pointer over the word "Display" and release the right mouse button. The display of the buffer contents begins at once.

(NOTE: Should you actually try this example with live data, you can control the buffer display by pressing the SPACE BAR to pause, any key to resume, and/or the ESC key to abort display. When the display is complete, OnLine! will wait for you to press a key on the keyboard before shutting down the display window and returning to the program.)

The principle here is that the program leads you through the menus, gradually isolating what it is that you want. This avoids the huge onscreen menu with a confusing array of options and its equally difficult to live with cousin, the ubiquitous command prompt (which gleefully beeps and buzzes at you when you enter a command incorrectly).

However, there IS value in being able to get to certain often user functions with a single keystroke. Sometimes things begin happening rapidly when you are online with a service, and using the mouse menus can get cumbersome. This is why OnLine! supports the Amiga-key sequences below:

Key	Action
E	Examine current settings and status
H	Hangup the telephone (turn off DTR signal)
R	Receive file with current protocol
S	Send file with current protocol
O	Open ASCII capture buffer
C	Close ASCII capture buffer
T	Terminate ASCII capture buffer transmission
G	Begin ASCII capture buffer transmission (Go)
P	Printer echo on
Q	Quit echoing characters to printer

Remember, to select one of these functions, hold down the RIGHT Amiga-key (the "open" A symbol) and type the letter of the function. You also have three special key sequences assigned to the ALT key (left of the space bar). These are:

ALT-C	Transmit 250ms of "break"
ALT-L	Engage learn mode for auto logon scripts
ALT-K	Engage learn mode for other script commands

The latter two will be discussed later, when we cover the use of OnLine!'s script language. ALT-C (holding down the ALT key and typing "C") will cause OnLine! to send 250ms of "break". If you continue to hold this down, the break will sustain as the key repeats. This is a signal required by some systems to signal that a request should be interrupted.

NOTE: If you do not know why you would want this, don't worry. This probably means that you don't need it. Those of you who WANT to send a break, now you know how.

Using OnLine!'s Requesters

Whenever OnLine! needs some extended input (other than "I want THIS menu option"), the mouse menu system becomes impractical. You must now be prompted for the information from the keyboard. Let's take time here and discuss these, since they are so widely used within the program.

OnLine! actually has several kinds of requesters. There is a standard line input requester, a special Archive requester and a special Macrokey requester. The latter two will only appear when you select an "Archive" option on one of the menus or when you select the "Macrokeys" option of the Tables menu, respectively. The standard line input requester,

however, will pop-up whenever OnLine! needs keyboard input from you.

These line input requesters will all appear and function in the same manner. There will be a single line within the requester and the ability to select "Ok" and "Cancel" at the bottom.

Sometimes the requester will already contain information. This happens when you have previously set information, such as a phone number, or when OnLine! is making a suggestion for input, such as in the case of the first time you open a capture buffer (more on this later). If the information in the requester is correct, you can select "Ok" and proceed.

If you have selected this option in error, do not be overly concerned. You can easily abort the request by selecting "Cancel". (When selecting something within a requester, point to it with the mouse and press the LEFT mouse button.)

If you need to enter information into the requester, or edit what information is already there, you can do so by selecting the input line itself. You will know when you have successfully done this, because a cursor will appear in the line. If there is some information already there, the cursor will appear at the end of this information. The keyboard is now active for input. Type in the needed information. You may use the arrow keys to move the cursor about, with the shifted arrow keys moving the cursor to the beginning and end of the line. The Delete key deletes the character under the cursor and you only need to type to insert new characters. The Backspace key will move the cursor one position to the left and erase the character that was there.

When you have completed your entry (or edit), press the RETURN key. It will not be necessary for you to also select "Ok" at this point. (As a matter of fact, the requester will disappear when you press RETURN. Its job is done.)

The Archive requester is a special interface that allows you to load and save files while running OnLine!. When you select an Archive option on one of the menus where it appears (Service, Capture, or Script), OnLine! will turn on the disk drive for a moment and then display the requester.

At the top of the requester, there is listed the directory being displayed and the pattern searched for. In the case of terminal and script files, the patterns ".trm" and ".scp" are added to the end of the filename you enter automatically when you store them. These patterns are then searched for on display. If you are only concerned with terminal files, you will not want to see any other files. This pattern is not used on Capture Archive, where all files are displayed. The directory indicator will reflect "CURRENT" if no specific directory was selected.

Underneath this header will be displayed a list of filenames and, optionally, comments. Positions not filled with a filename are marked with a line of dashes. If no files in the current directory match the file pattern, the screen will be empty. The first filename will automatically be selected for you when the requester is displayed. If you wish to select another, point to either the name or comment with the mouse and press the left mouse button. The new file will be highlighted.

Across the bottom of the Archive requester are the various functions you can perform while in this requester. They are:

Resume Get Replace Store Delete Dir Up Dn

Resume is used to exit back to OnLine! without doing anything. Get loads the selected file. Replace is used to overwrite the selected file with the current information (Service Archive would overwrite new terminal settings, Capture Archive the contents of the ASCII capture buffer, etc.). Replace will prompt for verification before overwriting a file. Store is used to create a NEW file on the disk. When you select Store, a line input requester will prompt for the filename

and optional comment. Filenames can be up to 20 characters in length and comments up to 30. Delete removes the file from the diskette, prompting to make sure you REALLY want to remove this file first. Dir is used to display a new volume or directory. When you select Dir, a line input requester will prompt you for the new drive/directory information. Up and Dn are used to page forward and backward when the list of files is larger than one screen can show.

The third type of requester is a Macrokey requester. This brings up a list of ten lines. On each line, you may enter a string of characters to be generated when the proper Function Key (at the top of the keyboard) is pressed. It can be very useful to have often typed commands on a single keystroke. This special requester can essentially be treated as a collection of line input requesters. Point to the line you want to modify with the mouse and press the left mouse button. The cursor will appear on the line and you may edit it. (Your editing options are described above.) You also have a set of "special" characters that will perform useful functions for you. These are:

| **Vertical bar.** (Also known as Shift-Backslash.) Wherever this appears in the macrokey, the program will send a carriage return. Place it wherever you would normally press RETURN.

' **Apostrophe.** This will cause a one second delay.

' **Reverse Apostrophe.** This engages the slow mode. All characters following this, until the end of the macrokey or another reverse apostrophe turns it off, will be sent at "typing speed". Useful when the host computer can't keep up with full speed transmission.

- **Tilde.** This "links" two macrokeys. It should be followed by the number of the next key to transmit. For example, if you wanted to send more data than would fit on one macrokey, you could set #1 with a link to #2. Then, F1 would transmit BOTH macrokeys.

^ Carat. Followed by one of the standard characters for ANSI control sequences (A-Z, etc.), this will transmit special values. For example, ^P sends a control-P and ^[sends an ESC.

Across the bottom of the requester are your options for when you have completed your editing. They are:

Ok Cancel New

Select Ok to accept the macrokeys as you have just edited them. The requester will be removed. If you select Cancel, all changes you have made since the requester first appeared will be eradicated. (Sort of a “whoops” button.) If you select New, all the current macrokeys will be erased, giving you a “clean slate” to work with.

With a little common sense, OnLine!’s requesters can be handled without any difficulty. On each requester there is a collection of possible actions. Once you satisfy the requester, normal operation of the program will resume.

Navigate about the program for a bit. When you feel confident to begin, we’ll start setting up for that first phone call

Setting the Phone Number

Now that you have learned about the general methods of talking to your OnLine! and have been introduced to its terminal files, it is time to begin configuring the individual elements in preparation for connecting up.

The first item of business is to tell OnLine! where to call. Press the right mouse button to display the command menu. Select Phone. For the time being, we will deal with the primary phone number only. Select Primary. This will bring up a requester that asks for the primary phone number. Enter the

phone number for the service you intend to call and press RETURN.

Setting Communication Parameters

Before proceeding, take note of the current power up default settings (press Amiga-key E or select Service Examine). They may be already set just the way you need them. Practically all of your commercial information services will use the same serial settings (7 bit word, even parity, 1 stop bit), while the majority of the local BBS systems will use settings more compatible with file transfer operations (8 bit word, no parity, 1 stop bit). The baud rate (bits per second) is the only usual modification.

All dial up services, whether your local bulletin board or a national information network for lawyers, will use one of the above two combinations. With a 50-50 chance of getting it right, it is not too difficult to get connected up. If one setting does not seem to work properly (usually you'll get "garbage" on your screen), try the other.

Baud rates are usually either 300 or 1200, although 2400 is gaining popularity. The power up default baud rate is 1200, but can easily be changed. OnLine! supports baud rates from 300 to 19200 bits per second.

Commercial Services

Word Length: 7
Parity: E
Stop: 1

Local bulletin boards

Word Length: 8
Parity: N
Stop: 1

If the service you are using wants you set for 1200 bps, a 7 bit data word, even parity and 1 stop bit, you can proceed to actually connecting with the service. (Select Resume to

remove the status window.) If it is necessary to change any of the COM settings, follow the procedure below.

Press the right mouse button to display the command menu. Select COM. This displays the menu used to adjust your communications parameters. You will see selections for Baud, Word, Parity, and Stops. You may select any of these that need alteration and you will be presented with a final menu that holds your possible choices. Select one. The check mark should move to reflect your new choice.

These alterations to the serial parameters (and the phone number set previously) will all be stored as part of a terminal file, eventually. This means that you should only have to do this work once for each service.

Check Your Work

Examine the OnLine! settings. Make sure the phone number is correct and the COM settings are appropriate for the service you are about to contact. If there is a discrepancy, go back over the last few paragraphs to correct the setting.

Saving Settings in a File

If you “store” these settings in a terminal file, all you have to do is “get” the terminal file from Service Archive the next time you want to call this service. Selecting Service Call then will automatically set the communication parameters and dial the phone number.

Name the file with up to 16 characters that will easily identify it with the service it accesses. File names are 20 characters in OnLine!, but the program will place a four letter (.trm) pattern on the end of the name. For example, if you are saving the terminal file you will use later to call

CompuServe, an appropriate filename might be something like "CompuServe".

Select Service Archive. When the requester appears, select Store. When the line input requester appears, in our example case, you would select the text area so that the cursor appears, type "CompuServe" and press RETURN. When the line input requester prompts for the optional comment, either repeat this procedure for a comment, up to 30 characters, and/or select Ok to proceed with the save.

Some Common OnLine! Customizations

Before you dial up and connect with the host computer, you may also want to take a few moments and consider customizing your OnLine! for this service and to conform to your personal taste. Specifically, these areas:

1. Window size. 80x23 versus 79x22.
2. Any desired command macrokeys. (Not needed for auto login, which will be handled by the script files.)

Window size. OnLine! defaults to a "framed" window with a sizing gadget at the bottom of the screen. This gives you an effective display size of 79 characters per line, with 22 lines. For many applications, you will want a full 80 character line. To accomplish this, select Other Window Borderless. This will turn the frame off at once. You will also get one extra line of text with the bottom border gone. You will NOT be able to size this window. To return it to normal size, select Other Window Normal. Any terminal files saved with the screen set in this manner will resume this configuration when loaded again.

Command macrokeys. Earlier, we described how to use the Macrokey requester. You will find Macrokeys very useful. Each terminal file can carry its own set of ten, so commonly

used commands or items such as your name and password can be stored for simple transmission.

Each of these areas is stored with the terminal file. You will only need to set them once for each service. (A "service" is a host computer that you call. A "terminal file" contains all the information peculiar to that service. A "script file" contains automated operation instructions. These can be linked to terminal files for automatic logon.

If you alter the window size or macrokeys and want to update the terminal file on the diskette, select Service Archive. When the Archive requester appears, highlight the file you want to replace by pointing at it with the mouse and pressing the left mouse button. Select Replace at the bottom of the requester. When asked for verification, select Ok. OnLine! will overwrite the old file with the new settings, including those changes you just made.

Connecting With The Service

Note: Before proceeding, make sure that the following conditions exist:

1. Modem connected and configured to your computer (per Modem Manual)
2. Phone line connected to modem
3. Phone line free (not in use)

Since the current terminal file, representing all of the changes (if any) you have just made, is active (as well as stored with Archive), you can initiate communication by having OnLine! dial the phone. Select Service Call.

You should connect up and have the service announce its presence or ask you for a reply (Name, password, etc.). If nothing happens, the service may be awaiting one or two carriage returns.

Error! Number Dialed but no Contact Made

(1) If the number is dialed but nothing happens, the most obvious source of trouble is a busy line. Try again by pressing ESC and repeating the call. Select Service Call.

IMPORTANT NOTE: When dialing a phone number, OnLine! will wait 30 seconds for the connection to be made and then it will abort. If you wish to abort this sooner than 30 seconds, as in the case of the busy signal mentioned above, press the ESC key. Selecting Service Hangup is NOT the proper way to do this! OnLine! will not see the Service Hangup selection until after the 30 second period. You must press ESC to abort a "wait for carrier after dial".

(2) By default, OnLine! programs Hayes and Hayes compatible modems to touch-tone dial. Some telephone equipment requires the older "pulse" (usually rotary dialing). If your modem supports this type of dialing, OnLine! lets you enter a Service pre-fix that may cause your modem to generate the proper pulses. Default is a prefix of ATDT which instructs a Hayes Smartmodem to dial with tone dialing. A Hayes will generate pulse dialing when an ATDP prefix is substituted.

Error! Contact Made but Garbled

If, after contact is established, extraneous characters appear on the screen mixed in with good text, this is probably

due to "line noise," a problem exaggerated with higher baud rates. Try changing to 300 baud to resolve the problem.

If, on the other hand, NOTHING is displayed correctly, the problem is most likely your serial parameters. If you using 7 bit word, even parity, then switch to 8 bit word, no parity. And vice versa.

Printing a Transcript of Your Session

The first time you make contact with a service, it is a good idea to have a printed transcript of the procedure that you can use to learn the system's commands. Further, the transcript will assist you in creating a script file for auto-logon purposes.

Creating a transcript is easy, assuming your printer is on-line with your computer. At any time you wish to have a print-out of your session, just select Other Printer On.

To turn off the printer, select Other Printer Off. If your printer is not online or cannot keep up with the data stream, you may capture the information to memory and/or a disk file for later review.

Capturing Text from a Service

Recording information you locate within an information service is most valuable if it can be saved in the form of a disk file. The capture process lets you set aside a portion of your computer's memory (RAM) to temporarily hold information prior to storing it in an archive file. If you will be downloading more than 64K of data, you should use the "capture to disk" method of downloading information.

Capture to memory. The simplest form of data capture. All you have to do is tell OnLine! to open the capture buffer.

If the service you are calling will open and close the buffer at the proper times for you (with CTL-R to open the buffer and CTL-T to close it), you should also close the buffer before exiting to the terminal mode.

To do this, select Capture Open. You will be greeted with a line input requester asking you to "Enter capture storage size (K):" It suggests a size of 64K. If you wish to use this as a capture buffer, you need only select Ok. This will be the case when doing a memory only capture. If you might be downloading MORE than 64K of data, then you will want to edit this to larger value, or perhaps better than that, use a capture direct to disk (explained below).

Next, you are prompted for a filename. Since we are dealing with a capture to memory here, select Ok with no input. This allocates a 64K capture buffer to memory only. As soon as you return to the terminal mode (and any incoming data has been buffered up and is waiting for you to return), you will begin capturing data. As mentioned above, if the service you are calling will control the buffer open/close, you should select Close before exiting this menu.

When you return to the terminal mode, the status display will now reflect the capture buffer being open or closed as it occurs. Remember to store the capture buffer to disk with Capture Archive Store before exiting the program. (This does not apply in a capture to disk. Capture Close or CTL-T from the host will write the buffer to disk.)

It is also important to close the capture buffer after you have received the data you are seeking. Failure to do so allows data to continue streaming into the buffer creating an unwieldy collection of data and wasted memory.

Regardless of whether the capture buffer is Open or Closed, data stored in it can be viewed with the Capture View option. If the data begins to scroll off the window, press the space bar to temporarily halt the flow. Pressing the space bar again allows the data to continue to scroll. Press ESC to abort

and close the window. Press any key when the display is completed.

When data is safely in the capture buffer, (and the buffer closed) you can "archive" the information in a file. Select Capture Archive. When the requester appears, select Store. Answer the prompts in the same manner as you would if you were storing a terminal file, with the exception that OnLine! will NOT add a pattern to the filename this time. Use all 20 characters, if you wish. Once the data is archived, you can safely clear the buffer.

To get rid of the capture buffer (and its contents), select Capture Clear. The first clear command empties the buffer, the second resets the capture buffer. You would use this when switching from a 64K memory only capture to a spooled direct to disk capture (since you cannot attach a filename to an already enabled buffer).

While a capture buffer is allocated, you will not have to re-enter the size and filename information. Open simply re-opens the buffer with the last parameters.

Capturing to disk. OnLine! also allows you to capture data coming in from the service directly to your disk drive. To do this, select Capture Open. If you do not get the "Enter capture storage size (K):" prompt, then you must already have a buffer allocated. Make sure that any valuable data in it has been archived to disk and then eliminate it as described in the preceding paragraph. Repeat the command sequence. If you get the memory size prompt now, you are ready to proceed. Replace the "64" with a "1" (for a 1K buffer). Press RETURN.

At the filename prompt, enter the name of the disk file (including path, if needed) that you wish to store the captured data in. As soon as you return to the terminal mode, OnLine! will begin spooling information to that file, in 1K blocks. If the

filename you specify already contains information, the new information will be added to the end.

If the service you are calling is going to control the capture buffer, again, select Close before leaving this menu. When you return to the terminal mode, the status display will reflect the capture to a file. Selecting Service Examine will reveal what file is currently being spooled to.

Each time the 1K buffer fills up, OnLine! will write it to the disk file. OnLine!'s automatic flow control will send an XOFF to the host, if the program gets too far behind on incoming data. When you are through capturing, select Capture Close. This will write the contents of the final buffer (even less than 1K) to disk. If you want to make sure that further CTL-R characters from the host do not re-open the buffer, select Clear twice before exiting this menu.

Sending a Text File to a Service

There are two methods of file transmission offered by OnLine!. Which you make use of depends on what sort of file you want to transmit and how large it is. The two methods are "text buffer send" and "protocol transfer".

Text buffer send. This method will work with text files ONLY. The file is loaded into the OnLine! capture buffer and this buffer is transmitted to the host one character at a time. To use this method, you would select Capture Archive. A list of files would be displayed.

Highlight the filename and select Get. This loads the text file into the capture buffer from the disk. If you want to verify this, you can use the Capture View command to examine the buffer.

The next step is to prepare the host service to receive the file. This varies from service to service. Some services will send you a prompt when ready for the next line of the file.

Some services will control your output with XON/XOFF flow control. And still others have no control over the text file upload, forcing you to slow down your transmission to typing speed.

OnLine! will handle any or all of these. Select Capture Send. You'll see the following menu:

Stop
Go
C-delay
L-delay
Prompt

Stop. This command aborts the transmission of the capture buffer. This can also be accomplished with Amiga-key T. Transmission is immediately suspended.

Go. Once the various prompts and/or delays are set properly, this begins sending the buffer. You will be returned to the terminal mode so that you may observe any echo of the transmission. This can also be accomplished with Amiga-Key G.

C-delay. This command sets a delay between EACH CHARACTER of the file as it is sent. Even with the "worst case" hosts, you should eventually be able to slow down the transmission sufficiently with this command to be able to upload a file. Values range from a 300 millisecond delay at "10" to over 6 seconds at "255".

L-delay. If the host can accept individual lines quickly, but requires a delay after each carriage return to prepare itself for a new line, this command lets you pause between LINES instead of CHARACTERS. The delay values range approximately the same as C-delay.

Prompt. Hosts that send you a prompt string or character can be handled with this command. Select Prompt and OnLine! will ask you what the prompt is. When you enter this character or string of characters, press RETURN. As soon

as you select Go, OnLine! will send the first line of the buffer. It will wait on the prompt before sending any subsequent lines. (The procedure here assumes that you waited until the host transmitted the first prompt before starting this whole process.)

This method of file transfer is excellent for uploading messages that have been prepared "offline" with a word processor. You should note, however, that few message editors like blank lines. If you want to space out the text in your message, you should use a line that holds a period and a carriage return.

Protocol File Transfers

Protocol transfer. Binary files (files that contain values outside of the ASCII character set) require a protocol transfer. We also recommend that text files larger than 64K use a protocol transfer to prevent line noise from corrupting the file's data. These protocol transfers send the file's data in blocks of varying sizes (depending upon the protocol) and will detect any errors that take place during the transmission, re-sending the block if necessary. OnLine! supports three such protocols. XMODEM/checksum (which is standard XMODEM), XMODEM/CRC, and Hayes Verification Protocol (for SmartCom compatibility). One of these should be supported by any system that you call.

OnLine!'s XMODEM protocol is "relaxed" for use with packet networks such as TymeNet. This term indicates that the critical timing loops have all been altered to give the data blocks sufficient time to be transmitted through the built-in network delays.

To select a protocol, select Transfer Protocol. The menu that appears will allow you to choose a protocol. This will be saved as part of a terminal file, so you can actually have a different "default" protocol for each service in your list. Once you have selected the protocol, you may either start the

receive or transmit at that time (with the Receive and Send options), or return to the terminal mode. Later, the Amiga-key R and Amiga-key S sequences can quickly start the protocol receive or send.

When prompted for the file to receive or send, you should type in the filename (including path, if it is in a different directory) and press RETURN.

A pop-up window will appear and inform you whether you are sending or receiving and what the filename is. When sending a file, OnLine! will tell you the number of blocks to send and estimate how long this should take at your current baud rate. (This does not take into account errors and re-transmissions, however.) To abort a protocol file transfer, select the close gadget in the upper left hand corner of this window.

With OnLine!'s powerful uploading and downloading capabilities, the door is opened into a new world of software. Hundreds of local bulletin boards will soon hold many of free or reasonably priced "shareware" programs that will expand your Amiga's functionality a thousandfold. The ability to capture text to memory or disk means you can store messages and information from services like CompuServe or The Source and peruse them when offline (and not compiling a huge bill!).

Conclusion

This concludes the first part of the OnLine! tutorial. The next part discusses creating script files for automatic log on and other purposes. Following that is the Reference Manual and the detailed script file commands appendix.

By no means was this supposed to be a complete primer on telecommunications. The ONLY way to learn the ropes is by "doing". As you use OnLine!, you will have more

questions. We hope these are answered by the command reference section. If not, you can turn to the bulletin board system or MSS Tech Support. But if we have given you enough information to encourage you to get started, then we've accomplished our goal. Welcome to the world of online computer users!

Chapter Two

Automating OnLine! Communications

In this part of the tutorial, we will discuss how you can automate your OnLine! communications sessions. Part of this automation is the storage of terminal files that simplify dialing up and connecting with host services. Another major part of this automation is the ability to create script files. These script files, which are actually programs in the script language, let you have OnLine! interact with the host service exactly as you might. With script files, OnLine! can automatically respond to prompts with your name or password, open a capture buffer, turn on the printer, and dozens of other tasks. Of course, auto logon is the major part of what script files are used for, but they can also take over many other tedious tasks.

There are two ways to create one of these script files. First would be with OnLine!'s "learn mode". In this technique, you simply log in to the service normally, and whenever faced with a prompt that OnLine! will have to respond to later, you press one of the two learn mode keys. (Either ALT-K or ALT-L.) The ALT-L initiates a simple "prompt-response" sequence, while the ALT-K sets up a "prompt-any other script command".

The first thing that happens when you press a learn mode function key is OnLine! will display the last 15 characters received and ask you to edit these into the prompt you want the program to wait for next time. If you want the entire 15 characters, just select Ok. If the prompt is not as you would like it, perhaps there are some extra characters in the prompt due to line noise, you can edit the prompt to clean it up and then press RETURN. The normal editing keys are functional here. Once you have pressed RETURN, what happens next depends on which key you pressed. If you pressed ALT-L, OnLine! only wants to know what the reply should be to this

prompt. Type it in, and remember to put a vertical bar (shift-backslash) wherever you want the program to send a carriage return (i.e. at the same spot you would press RETURN when typing it in manually).

If you had pressed the ALT-K instead, OnLine! will prompt you for the script command you would like to use. As opposed to the ALT-L, which types in the script command "Reply" for you and is only interested in the strings to wait for and respond with, using the ALT-K assumes that you know how the script commands are organized and wish to use a specific one. (NOTE: Should you press the ALT-K when you meant to press ALT-L, you could enter Reply "string|" and achieve the same results.)

In both cases, OnLine! will remember the prompt and the action that you want taken (whether a simple response or something more involved). OnLine! will also execute the indicated script command immediately. This is done so that you do not have to tell the program to remember the prompt, enter your response, and then enter it in to the host again. OnLine! would, for example, immediately send your name (or open the capture buffer, or whatever you have instructed the program to do).

Now, when you have completely logged in to the host service, you still must Archive this script file to disk. To do this, select Service Archive. When the list of terminal files appear, select the name of the terminal file currently loaded and select Replace. When OnLine! saves a terminal file, if there is a script file loaded into memory at the time, it is saved using the same 16 character filename you use for the terminal file. OnLine! will add the pattern .scp to the filename. The next time that this terminal file is loaded, the script file will ALSO be loaded. OnLine! will automatically

begin executing the script file when carrier becomes active (i.e. you make connection).

The second method for creating a script file would be to use a word processor or text editor. A program such as Ed (the editor included with AmigaDOS) will do nicely.

This would also be the proper method for editing a script file. In the case where a system changes a prompt or adds a new one, you would need to modify the script file. Remember, this is the same filename as the terminal file, except with the .scp pattern in the filename. The learn mode is useful for creating quick auto logon scripts and for those who have not yet learned OnLine!'s script language, but to get the most out of OnLine!, you will need to learn how to use the full range of script file commands.

Now let's take a more detailed look at scripts overall, with some examples.

What is a "Script File?"

A script file is a list of prompts and responses that guide the interaction between OnLine! and a remote "host" computer. In its simplest form, a script consists of a "dialogue" with "Wait strings" indicating a prompt from the host that OnLine! "waits" to receive. In addition, there are "Replies" that OnLine! sends in response to the wait string prompts. There are also a number of "script commands" that control the internal operation of your computer and manage

the execution of the script file. (The complete list of script commands is detailed in Appendix A.)

As mentioned above, there are two ways to create a script file. You can write it using Ed or let OnLine! help you write it with the "Learn" facility.

The following is an example of a script file that logs onto THE SOURCE via TELENET. (All codes and numbers have been changed for this example.)

```
wait delay 50
reply "|"
wait delay 5
reply "|"
wait string "TERMINAL="
reply "D1"
wait string "@"
reply "C 30124|"
wait string "THE SOURCE |>"
reply @1
```

A transcript of the example script file.

The following represents a transcript of the auto-logon procedure controlled by the script file above. Only the numbers and codes have been changed.

```
TELENET 303 15C
TERMINAL=D1
@C 30124
301 24 CONNECTED
Connected to THE SOURCE
>ID QV4457 BDYIO8
QV4457 (user 34) logged in Monday, 26 May
84
20:53:24.
Welcome, you are connected to THE
SOURCE.
Last login Friday, 23 May 84 15:20:16.
(C) COPYRIGHT SOURCE
TELECOMPUTING
CORPORATION 1984.
Now at 1200 baud, an extra value service.
->
```

Controlling scripts

A script can log you on, search for data, capture and save it, and disconnect. A complete list of script commands (located in the command section) can be used to create complex telecommunications sessions. You can even have one script file that logs you into the service and three or four more that automate repetitive tasks DURING the connection. These could be loaded separately with the Script Archive command sequence and started with Script Go.

When you store a terminal file, any script present in memory is stored under the same filename with the .scp pattern added to the filename (instead of .trm). A "link" to the terminal file is established and when you next load the

terminal file, the script file will load automatically and take over the operation when connection is established.

Selecting Script Suspend will temporarily suspend the execution of the script. Selecting Script Resume resumes the execution, so it is easy to put the process on “hold”. This might be useful if you wanted to prevent the script file from taking you into your desired database on a service until you have read the new opening bulletins. Basically, this is a “wait a minute” command. The keyboard is ALWAYS active in OnLine!, whether there is a script file running or not. If a service presented you with a new prompt, you can respond to it manually without interrupting the script. Remember to return and edit the script file later, though.

Creating an Auto-Logon Script Using “Learn”

If you are confident that you know every prompt and response associated with a particular service, you may write the script using any text editor that creates normal ASCII files. Otherwise, you should use LEARN mode to create the script, and a text editor to optimize and enhance it later.

To follow the creation of the example above, first a terminal file must be activated, including the dialing of the phone number. Remember that for this example, we are using an access network known as TELENET. When contact is made, your modem may flash a “CONNECT” prompt onto the screen. With some modems, you will only know you are in contact when the host system sends you a prompt.

In this example, two carriage returns are sent, the first a couple of seconds after the signal is received and the second just after that.

```
wait delay 50
reply "|"
wait delay 6
reply "|"
```

These commands would be added later with Ed, since they must be sent out without any prompt from the host. So, for the first session, you would press RETURN twice manually.

With that, a prompt appears:

```
TELENET 303 14C
TERMINAL=
```

"TERMINAL=" is your prompt so press the ALT-L. You will see a window at the bottom of your screen with something like this:

```
Wait string: 4C|\|\TERMINAL=
```

OnLine!'s learn facility reproduces the last fifteen characters received. To "clean it up," you can edit the string to leave just "TERMINAL=". Now press RETURN. The requester changes to ask:

Reply:

In this case, the terminal type is D1. To include a carriage return in the script, add a vertical bar (|).

Reply: D1|

Now, when OnLine! encounters "TERMINAL=", it will respond by sending "D1 RETURN". When you press RETURN after entering the reply, the string you just entered will be both transmitted and remembered, and you will return to the terminal mode.

Eventually, another prompt will appear on the screen. With each prompt, follow the same procedure. Press the ALT-L, edit the prompt, press RETURN. Type in the reply, using "|" as a carriage return, if one is required, and press RETURN to proceed.

After you have answered all the prompts necessary to log in, you can view the script with Script View. If the script starts to scroll off the screen, press the space bar to stop it. Press it again to continue.

Other script commands include conditionals such as IF and WHEN. And script files can transfer control to other script files. As a result, different script files can be "linked" and called up, depending upon the conditions present during the communication session.

Remember, to insert a script command instead of a simple response, press ALT-K instead of ALT-L. Again, you will be presented with the "wait string," but when you press RETURN, instead of a "reply," you can issue a "Script cmd:"

(which is any of the valid script commands listed in Appendix A).

An Explanation of the Status Window

One of the most common actions you will make is looking at the status window to ascertain your current settings. This is accomplished when you select Service Examine.

A sample status window display:

Service NOT STORED		Script NO	
Primary		Retries 0	
COM Settings		Capture settings	
Baud 1200	Parity EVEN	Size 0	Status CLOSED
Word 7	Stops 1	Free 0	Used 0
		File NONE	
Other settings			
Print OFF	Protocol XMODEM-STD		
Echo OFF	Terminal TTY		
CR+LF ON	Duplex FULL		
			Resume

Service. This displays the name of the current terminal file. This is set whenever you load or save a file with Service Archive Get (or Store). If the words "NOT STORED" appear, this is an indication that the settings currently in effect have

NOT been stored in a permanent file. You should do so at once to avoid possibly losing all the changes you have made.

Script. This tells you whether there is a script file loaded currently. Be careful. If this is set to "YES", when you archive a terminal file, it will store a script file with the same name. This could overwrite your auto logon script if you have loaded a subsequent script file since logging in. (If you HAVE, you should re-load the auto logon script after making changes to your settings and BEFORE you replace the old terminal file. This way the auto logon script gets saved out again, unchanged.)

Primary. The primary telephone number. This is the first number OnLine! will attempt to dial when the terminal file is loaded. If there is nothing set here, OnLine! will not attempt an auto dial when it loads the terminal file. There is also an alternate number that may be stored with a terminal file. The alternate number is dialed when no connection is made at the primary number (within the specified number of retries).

Retries. This contains the number of times, between 1 and 99, that OnLine! will try to auto dial the current phone number before giving up. If there is both a primary and an alternate phone number set, OnLine! will try the primary phone number FIRST for the COMPLETE number of retries and then do the same thing with the alternate number.

Baud. Line speed in bits per second. Set this to match the host service you will be calling.

Parity. This indicates the status of the parity bit. This bit, which follows the 7th bit in the data word, is used as a form of error detection on some systems. If parity is EVEN, the

parity bit will be set when the number of set bits in the data word is odd (making the total number of set bits transmitted even). If a bit gets flipped during transmission, the two will not match. A setting of ODD is just the inverse of this. If parity is set to NONE, parity error checking is not used and all 8 bits of the data word affect the character value (this is used to transmit graphics and the like).

Word. The word length in bits. This refers to the number of bits used to make each character. This should also be set compatible with the host service you will be calling. 7 bits is the minimum required for an ASCII service, so "7" and "8" are the most common settings. If you have a 7 bit word length, almost 99% of the time you will use EVEN parity checking. An 8 bit word will usually be combined with parity checking set to NONE.

NOTE: A setting is 7 bit word with no parity checking is not really valid. Most modems and services will tend to act strangely if this is the setting you end up on (things will SEEM to work), so checking for 7 bit word with no parity is often a good place to start when troubleshooting.

Stops. The number of stop bits. This will almost ALWAYS be set to "1". Your only other choice is "2", and this is rarely used. However, it is possible that some system may require 2 stop bits, so be sure and check this parameter when finding out how to configure OnLine! for some particular service.

Size. This tells you how large, in bytes, the currently allocated capture buffer is. Remember that you set the buffer

in kilobytes and this display is in bytes. A 64K buffer is 65535 bytes long.

Status. This tells you whether the capture buffer is currently OPEN (what comes in on the display is captured) or CLOSED (received data is NOT captured at all). If there is a buffer allocated (i.e. Size is not equal to "0"), a CTL-R from the host service will open the buffer and a CTL-T will close it. If there is no overflow to a file engaged, the "File" parameter will be set to NONE.

Free. This is the amount of capture buffer space, in bytes, that is free and remaining to be used. When this is filled up, if there is a filename set for an overflow to disk, OnLine! sends an XOFF, writes the buffer, and sends an XON to resume. If there is no filename attached to the buffer, the data capture simply ceases.

Used. This displays the number of bytes in the capture buffer currently filled with download data.

File. This is the optional capture to disk filename. If this is set, the capture buffer is automatically stored on the disk. For details, you can read the "Capturing text from a service" section in the previous section of this tutorial. This will be "NONE" if the capture to disk is not used.

Print. This reflects the status of the printer echo. If this is "YES", each character displayed on the video will also be printed on the line printer. (Make CERTAIN your printer is online and ready!) If this is "NO", incoming data is displayed on the screen only. This is toggled with the Other Printer Enable (or Disable) command sequence.

Protocol. Each terminal file stores which of the three supported error detecting and correcting file transfer protocols (XMODEM, XMODEM/CRC, and Hayes Verification Protocol) to use with the host service when a protocol transfer is selected. This will be "XMODEM-STD" for normal checksum XMODEM, "XMODEM-CRC" for XMODEM with the CRC option, or "HVP" for the Smartcom compatible Hayes Verification Protocol.

NOTE: The Hayes Verification Protocol stores files on disk with an EXACT end of file. Two Amiga owners using OnLine! to send programs between their computers will find it useful to employ this protocol.

Echo. If you are connected with someone else (chatting computer to computer), and the terminal program they are using can operate ONLY in full duplex, they cannot see what they are typing. If you turn echo ON with the Other Echo Enable command sequence, all characters received by OnLine! will be echoed back to the other computer, which lets the other person see what they are typing. OnLine! does not need this itself, since it is capable of half duplex operation.

Terminal. This indicates what terminal emulation is currently selected. OnLine! is capable of emulating several DEC terminals for connecting to a mainframe computer. "TTY" emulation is the standard dumb terminal used when specific emulation is not needed.

CR+LF. This indicates whether the special carriage return followed by line feed handling is engaged. If this is ON, OnLine! will display a line feed following a carriage return. When a host service sends you only carriage returns at the end of lines, this will prevent the text from being displayed all on

the same line. However, OnLine! is intelligent in this regard. When a line feed is received following a carriage return, and CR+LF is ON, the line feed is ignored. This makes the CR+LF setting of ON correct for any kind of system. You should only set this to OFF if it is important that carriage returns do NOT generate automatic line feeds (i.e. if the host service used carriage returns for special formatting and expected a line to be overwritten by new data). (NOTE: Any of the DEC terminal emulation settings will render this parameter ineffective. They have their own escape sequences which control this function.)

Duplex. This indicates whether OnLine! is operating in FULL or HALF duplex. In half duplex operation, characters that you type are echoed on to your screen locally by OnLine! and transmitted to the host service. This assumes that the host service will NOT be echoing back your input. Full duplex only transmits what you type to the host service and relies on that service to echo them back. Using full duplex gives you a simple form of error checking. If, for example, you type the letter "F" and the letter "G" gets echoed to the screen from the host service, at some point (either going there or coming back), the data got scrambled. This gives you the opportunity to backspace and correct it. Most host services are full duplex. The most common use for half duplex is when you are connected to another computer running a **TERMINAL** program (rather than a host), as will happen when you are transferring files, etc. In these cases, half duplex lets you see what you are typing. (NOTE: Remember, if the other terminal is full duplex only, you can turn OnLine!'s Echo ON.)

This status window display tells you, at a glance, all the important information about OnLine!'s configuration at the moment. Since getting the serial parameters matched correctly between the host service and the remote terminal (you), you

will find yourself referring often to this screen. This can also be called up from within the terminal mode with Amiga-key E. If you are already in the terminal mode, you may find it easier to check it this way than using the mouse menus.

Conclusion

This concludes the the OnLine! tutorial. In this section you were introduced to script files. For complete and detailed script file documentaion, refer to Appendix A of the manual.

In addition, we looked at the OnLine! status window, one of the most commonly used features of the program. We discussed each element of the window and reflected on why it is important to know the current configuration of the OnLine! program.

Next we'll examine each menu a command at a time in the OnLine! reference manual.

Chapter Three

Reference

Introduction

This portion of the manual is the OnLine! command reference section. In here we will examine all of OnLine!'s menu commands, one at a time. This is the spot in the manual you should turn to when you have a specific question ("Hmmm. I wonder what THAT menu item does?").

The commands are covered in menu order to make locating specific items faster. There is one common section for the Archive requester, since the subcommands are common between all occurrences of the Archive requester.

Command Descriptions

There are two primary modes in OnLine!: Command (when the menus are present) and Terminal where OnLine! allows you to interact with the remote computer system.

The Command mode is activated by pressing the right mouse button. You will know that you are in Command Mode because a command menu will replace the status bar at the top of your screen. While in Command Mode, communication will appear to cease; you will not be able to type (except for commands) and the screen will not display incoming data. However, incoming data will be captured and held until you return to the Terminal mode, at which time the screen will be updated (up to 1024 characters).

To select a command in the OnLine! command menus, you simply point to the menu name with the mouse pointer. This causes the pull-down menus to appear. While continuing

to hold down the right mouse button, move the mouse pointer over the menu option you want. To select this option, you can either release the right mouse button which selects the menu option and exits the menus or press the left mouse button which selects the option and remains in the menus.

If the menu option you select requires further input from you, OnLine! will bring up the proper requester.

Using OnLine!'s Requesters

Whenever OnLine! needs some extended input (other than "I want THIS menu option"), the mouse menu system becomes impractical. You must now be prompted for the information from the keyboard. Let's take time here and discuss these, since they are so widely used within the program.

OnLine! actually has several kinds of requesters. There is a standard line input requester, a special Archive requester and a special Macrokey requester. The latter two will only appear when you select an "Archive" option on one of the menus or when you select the "Macrokeys" option of the Tables menu, respectively. The standard line input requester, however, will pop-up whenever OnLine! needs keyboard input from you.

These line input requesters will all appear and function in the same manner. There will be a single line within the requester and the ability to select "Ok" and "Cancel" at the bottom.

Sometimes the requester will already contain information. This happens when you have previously set information, such

as a phone number, or when OnLine! is making a suggestion for input, such as in the case of the first time you open a capture buffer (more on this later). If the information in the requester is correct, you can select "Ok" and proceed.

If you have selected this option in error, do not be overly concerned. You can easily abort the request by selecting "Cancel". (When selecting something within a requester, point to it with the mouse and press the LEFT mouse button.)

If you need to enter information into the requester, or edit what information is already there, you can do so by selecting the input line itself. You will know when you have successfully done this, because a cursor will appear in the line. If there is some information already there, the cursor will appear at the end of this information. The keyboard is now active for input. Type in the needed information. You may use the arrow keys to move the cursor about, with the shifted arrow keys moving the cursor to the beginning and end of the line. The Delete key deletes the character under the cursor and you only need to type to insert new characters. The Backspace key will move the cursor one position to the left and erase the character that was there.

When you have completed your entry (or edit), press the RETURN key. It will not be necessary for you to also select "Ok" at this point. (As a matter of fact, the requester will disappear when you press RETURN. Its job is done.)

The Archive requester is a special interface that allows you to load and save files while running OnLine!. When you select an Archive option on one of the menus where it appears (Service, Capture, or Script), OnLine! will turn on the disk drive for a moment and then display the requester.

At the top of the requester, there is listed the directory being displayed and the pattern searched for. In the case of terminal and script files, the patterns ".trm" and ".scp" are added to the end of the filename you enter automatically when you store them. These patterns are then searched for on display. If you are only concerned with terminal files, you will not want to see any other files. This pattern is not used on Capture Archive, where all files are displayed. The directory indicator will reflect "CURRENT" if no specific directory was selected.

Underneath this header will be displayed a list of filenames and, optionally, comments. Positions not filled with a filename are marked with a line of dashes. If no files in the current directory match the file pattern, the screen will be empty. The first filename will automatically be selected for you when the requester is displayed. If you wish to select another, point to either the name or comment with the mouse and press the left mouse button. The new file will be highlighted.

Across the bottom of the Archive requester are the various functions you can perform while in this requester. They are:

Resume Get Replace Store Delete Dir Up Dn

Resume is used to exit back to OnLine! without doing anything. Get loads the selected file. Replace is used to overwrite the selected file with the current information (Service Archive would overwrite new terminal settings, Capture Archive the contents of the ASCII capture buffer, etc.). Replace will prompt for verification before overwriting a file. Store is used to create a NEW file on the disk. When you select Store, a line input requester will prompt for the filename and optional comment. Filenames can be up to 20 characters in length and comments up to 30. Delete removes the file from the diskette, prompting to make sure you REALLY want

to remove this file first. Dir is used to display a new volume or directory. When you select Dir, a line input requester will prompt you for the new drive/directory information. Up and Dn are used to page forward and backward when the list of files is larger than one screen can show.

The third type of requester is a Macrokey requester. This brings up a list of ten lines. On each line, you may enter a string of characters to be generated when the proper Function Key (at the top of the keyboard) is pressed. It can be very useful to have often typed commands on a single keystroke. This special requester can essentially be treated as a collection of line input requesters. Point to the line you want to modify with the mouse and press the left mouse button. The cursor will appear on the line and you may edit it. (Your editing options are described above.) You also have a set of "special" characters that will perform useful functions for you. These are:

| **Vertical bar.** (Also known as Shift-Backslash.) Wherever this appears in the macrokey, the program will send a carriage return. Place it wherever you would normally press RETURN.

' **Apostrophe.** This will cause a one second delay.

' **Reverse Apostrophe.** This engages the slow mode. All characters following this, until the end of the macrokey or another reverse apostrophe turns it off, will be sent at "typing speed". Useful when the host computer can't keep up with full speed transmission.

- **Tilde.** This "links" two macrokeys. It should be followed by the number of the next key to transmit. For example, if you wanted to send more data than would fit on one macrokey, you could set #1 with a link to #2. Then, F1 would transmit BOTH macrokeys.

^ **Carat.** Followed by one of the standard characters for ANSI control sequences (A-Z, etc.), this will transmit special values. For example, ^P sends a control-P and ^[sends an ESC.

Across the bottom of the requester are your options for when you have completed your editing. They are:

Ok Cancel New

Select Ok to accept the macrokeys as you have just edited them. The requester will be removed. If you select Cancel, all changes you have made since the requester first appeared will be eradicated. (Sort of a “whoops” button.) If you select New, all the current macrokeys will be erased, giving you a “clean slate” to work with.

With a little common sense, OnLine!’s requesters can be handled without any difficulty. On each requester there is a collection of possible actions. Once you satisfy the requester (provide it with enough information for OnLine! to perform the requested task), normal operation of the program will resume.

OnLine! Command Menus

The right mouse button produces this command menu.

Service Transfer Capture Script COM Other Phone Tables

Service Menu

The service menu controls those areas of OnLine! that affect the overall operation of the program (e.g. your list of defined services, examining the program’s status, etc.).

Your options are:

**Archive
Call
Hangup
Examine
Quit**

Archive

Manipulate terminal definition files. See “Operating OnLine!’s requesters”, above.

Terminal files contain all that OnLine! needs to know prior to connecting up with a service. Baud rate, word length, parity and stop bits, macrokeys, auto logon scripts, telephone number, window configuration, and more are all contained within these files.

When you load a terminal file that defines a service for OnLine!, the only further action you need to take is selecting Service Call.

Call

This Service menu option will send the auto-dial command to the modem and begin the wait for carrier sequence if there is a phone number currently programmed.

If no phone number is currently programmed, this command performs no function.

Hangup

This Service menu option drops the DTR signal for three seconds in order to hang up the telephone line. If your modem does not respond to the DTR signal transition, you may try typing ALT-C to send 250 ms of "true break". Some modems use this signal to hang up the phone line.

Quit

This Service menu option terminates OnLine! and returns to either WorkBench or CLI, depending on where you started the program from.

Transfer Menu

This menu controls the protocol file transfers. There are three situations in which using the protocol file transfer method is appropriate: 1) When the data you wish to send is too large to fit into memory, necessitating disk file transfer; 2) When the data you wish to send is not in ASCII, as machine language programs; and 3) When you wish to transmit the data with error detection and correction. Your options are:

- Receive
- Send
- Protocol
 - Xmodem
 - CRC-Xmodem
 - HVP

Receive

The Receive option tells OnLine! that you are about to receive the transmission of a file. Amiga-key R initiates the same process. OnLine! will open a requester asking for the name of the file to receive. Move the mouse pointer to the input area and press the left mouse button to select it. A cursor will appear. At this point, enter the filename and press RETURN.

OnLine! displays a window that keeps you informed as to the status of the transfer. At any time, you may abort the transfer by selecting the close gadget in the upper left hand corner of the transfer window.

Send

The Send option instructs OnLine! to begin sending a file. Amiga-key S initiates the same process. As with receive, OnLine! will present a requester asking for the name of the file to be sent. Respond in the same manner (select the input area with the mouse, type in the file's name, and press RETURN).

OnLine! displays a window that keeps you informed as to the status of the transfer. At any time, you may abort the transfer by selecting the close gadget in the upper left hand corner of the transfer window.

Protocol

The Protocol option produces a sub-menu that has on it several sub-options describing the file transfer protocols that OnLine! supports. Select one. (This becomes the new default protocol and will be used automatically with any protocol file transfer. Your default protocol is stored as part of a terminal

definitions file and thus can become different for each system you call.

Xmodem

This option selects the normal XMODEM checksum protocol. The XMODEM protocol is perhaps the most popular protocol in use among microcomputers.

CRC-Xmodem

This option selects enhanced XMODEM with CRC option. This provides an extra measure of error detection when used with host services that support it.

HVP

This option selects the Hayes Verification Protocol (HVP), which is compatible with Smartcom. It is especially useful when you need to transfer with a EXACT end of file.

Capture Menu

Data that is located in an information service data base has limited value unless you can retrieve and store it. A series of OnLine! functions will help in this regard. Your options are:

- Archive
- Open
- Close
- Clear
- Send
 - Stop
 - Go
 - C-delay
 - L-delay
 - Prompt
- View
 - Display
 - Printer

Archive

The Archive option allows you to change default drives, erase files, get files from storage or replace the contents of a file with more current data. It is also possible to store data captured in the capture area of memory in an archive file. See "Using OnLine!'s requesters", above.

Open

Before incoming data can be captured, either to memory or disk, the capture buffer must be allocated and opened. The Capture Open option handles this task. When you select Capture Open, you will be presented with the first of two requesters, this one asking you how much memory to allocate to the capture buffer.

The requester "suggests" 64K, and if you want to use this value, simply select Ok. Otherwise, you may select the input line and edit the value. Be sure and express your input in kilobytes instead of bytes (i.e. 64 instead of 65535). Press RETURN when you are done editing.

Although OnLine! is capable of allocating all of your Amiga's remaining free memory to the capture buffer, this is not usually wise. If you should discover that you want to run any other applications from the WorkBench while you are capturing data, they will need memory also. 64K is a good working size, and you should never need more than 128K. For capture sessions larger than that, we suggest strongly that you capture the data directly to diskette. This is one of the reasons why we do not automatically allocate all of your free memory to the buffer when you select Capture Open.

The other reason is that when capturing to a disk file, you want your buffer to be much smaller. We recommend 1K, but you should never use more than 10K. OnLine! fills the buffer, writes it to disk, clears it out, and repeats the process until (a) the host system sends a CTL-T or (b) you select Capture Close. The less it has to write to the disk each time, the faster the entire process will be. So, set the size of the buffer according to your needs and you'll proceed to the next.

The second requester asks you for a disk filename. This filename specifies which file OnLine! will write the data being captured into. If you are using a memory only capture, select Ok and proceed. If you want to capture directly to disk, enter the filename here and press RETURN.

When capturing to disk, be sure to select Capture Close before clearing the buffer or exiting the program.

When a capture buffer is allocated, but not opened, and the host system sends OnLine! a CTL-R, the capture buffer will be automatically opened. A CTL-T will close it. When you are connected to a host system that will do this, you may select Capture Close after Capture Open to leave the buffer allocated but closed.

The “Capture:” indicator at the top of your screen will reflect the current status of the buffer (Open, Closed, or Open/File, Closed/File).

Close

The Capture Close option will close the capture buffer (stop capturing data) and write any partial buffer to disk (assuming a capture to disk is in progress).

Be sure and use this command, when capturing to disk, before using Capture Clear or Service Quit. When capturing to memory only, use Capture Archive Store to write the contents of the buffer to disk, if you want to store them, before exiting the program or clearing the buffer.

Clear

To prepare for additional data capture, you can clear out the capture buffer (erase its contents). There is no verification for this, so use care when you select Capture Clear. Any data in the buffer is immediately erased.

The Capture Clear option also can be used to eradicate the capture buffer itself. This is useful when you want to change from a memory only capture to a capture to disk or vice versa. To eradicate the buffer, simply select Capture Clear while the buffer is empty (i.e. the Used: parameter reads “0”). If there is data in the buffer that is not important to you, you can select Capture Clear twice to accomplish the same task.

Send

Data in the capture buffer may be transmitted to remote sites with OnLine!. The Capture Send option initiates this process. A text file can be transferred from Archive to the capture buffer using the “Get” command. Capture Send works

with any ASCII file. For the transmission of binary files, or files too large to fit into the capture buffer, use the Transfer menu and one of the file transfer protocols.

Pacing Data Transmission

In some cases, your computer may overrun the remote computer by trying to send data too quickly. OnLine! supports X-ON/X-OFF handshaking, but in its absence, allows you to accept prompts from the host computer to pace the data transmission. Additionally, OnLine! lets you insert line and character delays to slow down the transmission of data (if the host cannot prompt you for each new line).

Stop

The Capture Send Stop sub-option aborts the transmission of the capture buffer. Use this in the case of an error during the transmission.

Go

Capture Send Go commences the transmission of the data in the capture buffer with any previously configured delay parameters. (You should set any prompts or delays PRIOR to selecting Capture Send Go.)

C-delay

Set character delay time (0 - 255). A character delay is a pause that is inserted between each character that the computer sends. Value range from a 300 millisecond delay at "10" to a maximum delay of over six seconds at "255". The C-delay option can be used to slow the pace down to about the speed of normal typing.

L-delay

Set line delay (0 - 255). A line delay is a pause that is inserted with every carriage return (line end). This is used when the host can accept the individual lines at full transmission speed, but needs a delay at the end of each line

to allow for scroll (or something similar). Values are essentially the same as with C-delay.

Prompt

Hosts that send you a prompt string or character can be handled with this sub-option. Select Prompt and OnLine! will ask you what the prompt is. When you enter this character or string of characters, press RETURN. As soon as you select Go, OnLine! will send the first line of the buffer. It will wait on the prompt before sending any subsequent lines. (The procedure here assumes that you waited until the host transmitted the first prompt before starting this process.)

View

The contents of the capture buffer can be seen with the Capture View option which displays the current contents on either the screen or the printer.

When data is being displayed, you may press the SPACE BAR to pause or the ESC key to abort. At the end of the buffer's data, OnLine! will pause and wait for you to press a key before resuming normal operations.

Display

Capture View Display prints the contents of the capture buffer on the screen.

Printer

Capture View Printer prints the contents of the capture buffer on the screen AND the line printer.

Script Menu

A script file can be used, in its simplest form, to automate the log-on process, but scripts can do much more. A

script allows you to "program" OnLine! to interact with a remote computer, much as if you were present to direct the program. Script files are actually programs in the script "language". The complete list of script commands follows the general command index in this manual.

Archive

Archive allows you to load or save script files using an Archive requester (see "Using OnLine!'s requesters", above). Your auto logon script files are linked to your terminal files and will be loaded automatically. Any subsequent script files that you require while online should be loaded with this Archive option.

Stop

This option immediately stops execution of the current script file. Select this option when some error has occurred making it necessary to abort the script.

Go

The Script Go option actually initiates the execution of a script file. Normally the script file will execute when carrier is detected following an auto-dial with Service Call. If, however, you wish to begin a script mid-session or without carrier present, Go will handle that.

When there is a script file currently executing, the "Script:" indicator in the status line will reflect "On".

Resume

The Resume option re-starts a script file that has been paused with the Script Stop option. When a script has been stopped, it can either be started completely over with the Script Go option or picked up where it was left off with Script Resume.

Clear

The Script Clear option erases the current script file from memory. This prevents the script from automatically executing when an off-to-on transition of the carrier detect signal takes place following a Service Call.

View

The Script View option permits you to view the current script file. While the script file is being displayed, the output may be paused with the SPACE BAR and aborted with the ESC key. When the display is complete, OnLine! will pause and wait for you to press a key before closing down the window and resuming normal operations.

COM Menu

The COM menu controls all of your serial parameters. Items such as baud rate, word length, parity check, and stop bits. In addition, you can perform a quick "reset" to the power-up default values for each of these (1200 baud, 7 bit word, even parity, 1 stop bit).

Reset

Reset to default settings. In the event that you have temporarily changed one or more parameters, they can be easily returned to the default setting. The Reset option returns all COM settings to the pre-set "system" default (1200 Baud, 7-bit word, Even parity with 1 stop bit).

Baud

Baud rate (bits per second) selection. It is essential that computers communicate with each other at exactly the same rate of speed. Speed is set in bits per second or baud rates between 300 and 19200. Most of your communication will be

300 or 1200 baud, but 2400 baud modems are becoming more popular as time goes by.

Remember, the higher the Baud rate, the less time is required to transmit a data file. 300 baud is approximately 30 characters per second, 1200 baud is 120 characters per second, and 2400 baud is 240 characters per second. Higher speed modems GREATLY reduce file transfer times.

Word

Word size select. Serial data is essentially a stream of bits flowing from the host computer to the terminal computer and vice versa. This bits are grouped together into the individual characters by OnLine!. Inside this character is the actual value for the character being sent. This is called the "data word". The number of bits that make this up is called the "word length". You may select either 7 or 8 bit word length. If you use a 7 bit word, you will use even parity check. An 8 bit word uses no parity check.

Parity

Parity error checking is common for evaluating data integrity. OnLine! allows you to use even, odd or no parity at all. The number of bits "set" (1 instead of 0) in each data word is totalled. If this value is even, and parity check is odd, the parity bit is set to make the total number of bits set in the data word. The reverse is true for even parity check. No parity check is used with 8 bit word lengths, since the 8th bit is part of the character and not available for parity checking.

Stops

In order for two computers to communicate properly, a method exists to define where one character begins and another ends, "synchronizing" communication on each individual data word; this is called "asynchronous". (As

opposed to synchronous, which uses a preset clock timer and transmits data at regular intervals.) Part of this method is the stop bits between the words. You can set either one or two. One is by FAR the most common (and the default value).

Other Menu

This menu contains various options vital to operating OnLine! that don't fit into any of the other menus.

Printer

This option toggles the printer echo on and off. When the printer echo is on, all characters received from the host computer are displayed on the screen AND printed on the line printer. Output to the line printer is done "raw", that is, without any interpretation by the Amiga. It is sent on a character for character basis. The sub-menu for this option lets you select Off or On and displays a check mark to tell you what the current status is.

You may also turn the printer on with the right Amiga-key P and off with right Amiga-key Q. When the echo to the printer is active, the indicator on the top status line reflects this.

Terminal

OnLine! allows your computer to emulate four terminal types. The default, 5th type, is as a so-called "dumb" terminal, in which case it acts much like a teletype. Hence, the designation: TTY.

OnLine! supports the popular DEC terminal emulation configurations: VT-102, VT-100 and the older VT-52.

When using any emulation modes, you should eliminate the screen border. OnLine! will dispose of the status bar for you. You now have the full screen width of 80 characters and the full height of 24 lines.

Not all features are emulated, such as the 132 column screen width or "smooth scrolling." However the alternate keypad on the VT100 and VT52 are available as are the four PFS (programmable function keys).

The PFS keys are accessible using the ALT key in combination with the F1,F2,F3,F4 keys. The arrow keys send the proper values and keypad emulation is supported. The Help key emulates the keypad's comma.

When you select the Other Terminal option, a sub-menu appears with the additional sub-options to allow you to choose your terminal emulation type. The ANSI emulation includes the ANSI standard escape codes, but not the VT-100 extensions to these.

Width

The Other Width option allows you to define a new screen width for video re-formatting. When this value is set to anything other than "0" (no re-formatting), all incoming carriage returns are translated into spaces and the first space received after the specified width is translated into a carriage return.

For example, the text is coming in formatted for a 40 column display. You set your video width to 72 (it is always a good idea to set this value for 8-10 characters less than your actual display width). All incoming carriage returns, usually in column 29, are made into spaces. Once 72 columns have been displayed, the next space is translated into a carriage return.

This causes a neatly formatted display on the Amiga's wider screen.

IMPORTANT NOTE: Be aware that this will cause some very interesting things to happen to formatted items like menus and message headers. Video re-formatting has only limited application. It cannot tell the difference between a menu's carriage return and the carriage return at the end of a message line.

Duplex

This option lets you select whether you want to operate in Full or Half duplex. Full duplex is when your characters are not echoed to the screen as you type them, but rather when the remote computer echoes them back to you. In half duplex, what you type is echoed immediately. This is often used when "chatting" between two computers.

Select the Duplex option on the Other menu and when the sub-menu appears, select either Full or Half. Most bulletin boards and information services will operate in Full duplex.

Echo

This option tells OnLine! to echo back to the remote computer each received character. This is useful when you are chatting with someone who has only full duplex capability (i.e. cannot configure to half duplex for a local echo of their own keystrokes). Please be careful never to engage this function when connected to a full duplex service. This produces an "endless loop" of echoed characters between your computer and the remote.

Select the Duplex option on the Other menu and when the sub-menu appears, select either On or Off. For most applications, Off will be the proper setting.

CR+LF

This important option controls how OnLine! handles its carriage return/line feed groupings. The official Amiga "end of line" character is a line feed (0AH). This is defined in the Amiga manuals as causing a home cursor to start of line and a cursor down one line. This is NOT, however, the normal action a line feed takes, nor is it what some host systems will expect you to do when sent a line feed.

Therefore, OnLine! returns the line feed to its normal function of moving the cursor down one line without changing its column position at all. A carriage return will perform a home cursor to start of line function.

However, with this return to standard function comes the old problem of what to do about the fact that some host systems will send carriage returns followed by line feeds at the end of each line and some services send only carriage returns. The CR+LF option takes this all in stride.

When this option is engaged, any line feeds received AFTER carriage returns are ignored. And all carriage returns display both a home cursor to start of line and cursor down one line function. This way, if the service sends you just carriage returns, the display is fine. If a service sends you carriage returns and line feeds, the display is still fine. (By the way, all of these manipulations affect ONLY the display. Captured data or printed data is exactly as it was received.)

If you should happen to connect with an Amiga BBS that sends only LINE FEEDS, you can also adjust easily for this. Leave CR+LF set On and use the translation table for the Display to make a line feed (0AH) into a carriage return (0DH). Since CR+LF is On and line feeds are displayed as carriage returns, the display is again fine.

Flow

This option engages or disengages OnLine! flow control. This flow control system assures that OnLine! will not lose data when communicating with systems at a higher speed than the display can be updated. The method used is XON/XOFF. An XOFF is sent when OnLine! gets far enough behind that there is a danger of data loss, and as the display catches up, an XON is sent to resume output.

Some host systems may not use XON/XOFF, or worse, may honor the XOFF but not the XON. In these cases, it is best to disable the flow control and use other methods to avoid data overrun, such as lowering the baud rate or using null characters at the end of lines.

Select the Flow option of the Other menu and when the sub-menu appears, select On or Off. The majority of the time, you will probably want this On.

Window

OnLine! offers some very powerful configuration options regarding the size and function of its display window. These include a full 80 character line, a 79 character line with the ability to "size" the window, and a special split screen mode with a local echo window useful in chatting or with CompuServe's CB Simulator.

Normally, OnLine! display window is 79 characters wide and 22 lines high. There is a sizing gadget in the lower right hand corner of the window that allows you to shrink this size, which is useful when you have OnLine! and some other task running at the same time. However, for some applications, a true 80 column display is preferable.

In these cases, you can turn "off" the window border. This gives you an 80 column by 23 line display. (For DEC terminal emulation, which REQUIRES 80 columns by 24 lines,

the status line will also be removed.) When the window is set up in this manner (80x23 with no border), the sizing gadget is also removed. You cannot alter the size of the window until you restore the border.

The condition of the window, 79x22 or 80x23, is stored in the terminal files and will be automatically set when the file is loaded.

The "split" window sets up a small local echo window at the bottom of the screen (this can be relocated or re-sized if you want). This window displays your keystrokes **ONLY**, separating them from incoming data. This is useful when you in some kind of electronic "conference", like CompuServe's CB Simulator, where what you are typing gets interspersed with incoming data from other locations, making it difficult to see what you are typing. There is no buffering with this window, what you type is transmitted normally. It is only an echo of your keystrokes in a protected space.

Select the Window option of the Other menu and when the sub-menu appears, pick either Full, Split, Borderless, or Normal. (Full negates the Split option.)

Phone Menu

The Phone menu controls the parameters used by OnLine! when dialing the telephone with your modem. OnLine! comes set up for the Hayes Smartmodem or compatible units, but you should be able to adapt the Prefix and Suffix options to adapt to almost any command driven modem.

("Command driven" is a term that indicates the modem accepts the transmission of ASCII characters from the software as commands for dialing. Some modems use logic signals, such as the RTS line, pulsed rapidly to dial. OnLine! will not

operate with these modems. An acoustic coupler modem does not auto-dial in the first place, so this is not a problem.)

OnLine! supports a primary and alternate phone number for each service. Also, OnLine! can be told to re-dial the primary number, if busy, a specified number of times before trying the alternate number (an equal number of times).

Primary

This option defines the primary phone number. Up to 40 characters can be specified per phone number. Pauses can be programmed as required for some long distance services by inserting "~". (This causes a one second pause in the output of the phone number.) Some modems (such as Hayes Smartmodem) will respond directly to a "," inserted into the phone number for this purpose.

When you select this option, a requester will appear that prompts you for the phone number. The primary number is displayed in the status box shown by Service Examine.

Alternate

This option defines the alternate phone number. As has been alluded to, this number is dialed after the primary number fails to get results. This should be a second number for the same service. If there are any re-dial attempts specified, these will all be tried on the primary **BEFORE** the alternate number is dialed the first time (which is then itself dialed for the specified number of times).

Retries

When calling systems that are frequently busy, it is useful to be able to instruct the program to automatically re-dial the phone number until it gets a connection. This menu option controls the number of times OnLine! will try.

When OnLine! sends the dialing command to the modem, it waits approximately 30 seconds for carrier to come active. If this happens, it automatically activates the script file loaded in memory at the time (if one is). If carrier does not come active within 30 seconds, OnLine! drops the DTR signal to hang up the modem (if it has not already given up on its own) and if there are any retries specified, it sends the dial command again and starts the whole process over.

Retries automatically affect both the Primary and Alternate phone numbers. The Primary number is tried the complete number of times before the Alternate is dialed once, and then the Alternate is tried for the complete number of times.

Prefix

This option sets up the string of characters OnLine! sends to the modem BEFORE sending the programmed phone number. This string defaults to "ATDT", which is the command to tone dial a Hayes Smartmodem, but can be easily edited to conform to the needs of another modem.

You have several special characters you can use here:

' Reverse apostrophe. This slows the transmission of the dialing prefix down to "typing speed" by delaying 250 ms. between each character. Useful when your modem cannot keep up with the dial command because of echo.

- Tilde. Causes a one second pause in the transmission of the dialing prefix.

| **Vertical bar** (shifted backslash). Generates a carriage return. Put this wherever you would normally press RETURN.

^ **Carat**. Sets up the NEXT character as a control sequence. For example, “^Q” will send a CTL-Q.

Suffix

This menu option sets up the string of characters that OnLine! sends AFTER the phone number. This string defaults to a vertical bar (carriage return), which is used by the Hayes Smartmodem and compatibles, but can be easily edited to conform to the needs of other modems.

The same set of special characters is available here as there are in Phone Prefix.

Tables Menu

This menu controls OnLine!'s translation tables and its keyboard macros. A translation table lets you changed any one byte value to any other one byte value. There are seven of them, covering all of the various system devices. These are excellent for altering or removing control values that may have undesirable effects. You can also use the Display table for the special line feed only compensation described under the CR+LF option of the Other menu.

A keyboard macro allows you to define the ten function keys at the top of the keyboard to transmit up to 64 characters at the press of a single key. This is very useful for programming commonly used commands of the different systems you call. You can save a lot of typing with these!

MacroKey

This option permits you to define the ten function keys at the top of the keyboard as macrokeys to transmit up to 64

characters at the press of a single key. When you select this option, you will see the MacroKey requester.

This brings up a list of ten lines. On each line, you may enter a string of characters to be generated when the proper Function Key (at the top of the keyboard) is pressed. It can be very useful to have often typed commands on a single keystroke. This special requester can essentially be treated as a collection of line input requesters. Point to the line you want to modify with the mouse and press the left mouse button. The cursor will appear on the line and you may edit it. (Your editing options are described above.) You also have a set of "special" characters that will perform useful functions for you. These are:

| **Vertical bar.** (Also known as Shift-Backslash.) Wherever this appears in the macrokey, the program will send a carriage return. Place it wherever you would normally press RETURN.

' **Apostrophe.** This will cause a one second delay.

' **Reverse Apostrophe.** This engages the slow mode. All characters following this, until the end of the macrokey or another reverse apostrophe turns it off, will be sent at "typing speed". Useful when the host computer can't keep up with full speed transmission.

- **Tilde.** This "links" two macrokeys. It should be followed by the number of the next key to transmit. For example, if you wanted to send more data than would fit on one macrokey, you could set #1 with a link to #2. Then, F1 would transmit BOTH macrokeys.

^ **Carat.** Followed by one of the standard characters for ANSI control sequences (A-Z, etc.), this will transmit special values. For example, ^P sends a control-P and ^[sends an ESC.

Across the bottom of the requester are your options for when you have completed your editing. They are:

Ok Cancel New

Select Ok to accept the macrokeys as you have just edited them. The requester will be removed. If you select Cancel, all changes you have made since the requester first appeared will be eradicated. (Sort of a “whoops” button.) If you select New, all the current macrokeys will be erased, giving you a “clean slate” to work with.

A different set of macrokeys can be stored with each and every terminal file.

Editing the translation tables

When you select the menu option for any of the following translation tables, a very special requester will appear. This is a large table of character values, displayed in hexadecimal (00–FF). On the right hand side of the requester are the Ok and New selections.

To change a value, point to EITHER of the character’s two digits (depending on which needs to be changed) and press the left mouse button. The digit you are pointing to will increment by one. Continue to press the left mouse button until the value has become that of the NEW character.

For example, if you wanted to translate all of the “A” characters to “B” on the Display, you would edit the Display table, point to 41 (the value for “A”), and press the left mouse button when over the right hand digit, making it 42 (the value for “B”). Whenever an A is sent to the display after that, it will be translated into a B.

When a character is translated from its original value, it will be displayed in a different color to aid you in quickly locating the translated characters.

When the table is completely edited, select Ok. This will return to OnLine!'s terminal mode with the new values in place. If you want to restore an ENTIRE table to its original values, select New. This re-draws the table with all values restored to normal.

Display

The Display table alters characters before they are displayed on the screen. There is one special use for this table not mentioned anywhere else. Normally, the backspace on the Amiga is non-destructive (when the backspace is sent, characters underneath the cursor are not erased). This is not desirable in many instances. To make OnLine! use a destructive backspace, translate BS (08) into a DEL (7F).

Printer

The Printer table modifies data as it is sent to the printer. This is another commonly used table. Many printers engage special features on control codes that may be contained in downloaded text. It is possible to filter those codes out here (change their values to 00).

Keyboard

The Keyboard filter modifies data from the keyboard. If you should need to generate a keystroke value that is not on the keyboard, you can translate an existing value to do it. The MacroKeys are filtered through the table as well.

From-capture

The From-capture filter table modifies data as it is being sent from the capture buffer during transmission.

To-Capture

The To-Capture filter table modifies data as it is stored in the capture buffer during download.

In-COM

The In-COM table filters data as it comes IN to the computer, before any other processing is done on the data. Sort of a “master filter”. If this table changes the value, OnLine! sees it as changed everywhere.

Out-COM

The Out-COM table filters data that is sent out over the COM port. Sort of the inverse of In-COM, it is also a “master filter” of sorts. No matter where the value is generated from (keyboard, buffer, etc.), if its value is changed here, it will be sent out as changed.

Conclusion

This concludes the OnLine! Reference Manual. Next is a command-by-command description of the script language.

Appendix A

Script Commands

Introduction

OnLine! provides a special “language” in which users may write programs that control the OnLine! program. These programs may be used to log in and out of various services, download data, or almost any task normally performed by an operator using OnLine!. These programs may be written with a text editor (such as Notepad or Ed), or by using the ALT-K and ALT-L keys (i.e. the “learn mode”). These programs add a GREAT deal of flexibility to OnLine!.

We will refer to these programs as “script files”. The script language will, as with any other language, allow you to create programs that are as simple or complex as the operator chooses to make them. Script files are not to be confused with terminal files; although they are linked together, they are separate and distinct items.

The largest distinction between terminal files and script files is that terminal files are special disk files that hold all the configuration information needed to set OnLine! up for a particular service. As a matter of fact, EVERY parameter that is user-configurable is saved in a terminal file. Included in this are such items as the telephone numbers (primary and alternate), COM settings, macrokeys, translation tables, and anything else included in the Default settings. Script files, on the other hand, are simply text files that contain a list of commands. They are program files to be interpreted by OnLine!. They do not directly alter OnLine! itself, like terminal files do. Term files will have the patter .trm, and script files .scp

If they are so different, then how are they linked together? This is a good question. When you Archive a terminal file, OnLine! checks to see if there is a script currently in memory. If there IS, then a script file bearing the same name, but with the .scp pattern, will be Archived, also. Optionally, there is an Archive command in the Script menu from which you may load and save script files completely apart from any other operation. There is a caution in this. If you were to have a terminal and script file combination set up for a particular service, you must be reminded of the fact that whenever you save the terminal file again, you also save the script file, including any "on-line" additions or deletions you have made. If you have made alterations to the script in memory and you wish to restore it to its previous state, be certain to use the Script Archive command to load the old file again before you Archive the terminal file.

There is a whole set of script commands that we will cover shortly. Several of these control items of OnLine! such as the printer or capture buffer, but most are useful only in operation of the script program. If you are experienced in programming with BASIC or some other high level language, writing script files should come easy to you. If not, you may edit the script file samples we have provided for you. There is more than likely one that is close to what you need.

But, in either case, please remember that we cannot write script files for you, nor can our Tech Support personnel be expected to help you write script files for any particular purpose. Programming in ANY language is an acquired skill, and we are not obligated to teach programming classes over the telephone. We can perhaps identify when you are using a command or statement incorrectly, but that is about the limit. An electronic forum for the exchange of script files between users will be provided.

What is a script file?

A script file is simply a disk file that contains a list of one or more of the legal script commands or statements. When loaded and executed, this program has the ability to control OnLine! completely, much the same as if an operator were stationed at the keyboard. Script files are executed line by line from the top of the file downward. You may append statements together on a single line with a colon separator (:), if you so desire.

Script files can look for prompts from the host and respond with the proper reply, or they can delay for a specified period of time and then perform a task. During the delay, they can optionally be checking for a string of text from the host, and if it arrives, perform another task. Script files can request input from the operator and act upon it, message the operator, branch within themselves and many other powerful items.

A script file, if loaded, will automatically begin executing when carrier is established. Optionally, you may use the Script Go command to manually start the script file running. The total number of script files you may have is limited only by the amount of available disk space you have.

Creating a script file

There are two methods creating script files. One is to use a word processor or text editor to write the files. This method gives you the ability to review and edit the files, and will often be used in any event to correct errors and make additions or deletions to the script. To use this method, you will either have to be familiar enough with the service being used to know what prompts it will send you and what responses you will send, or you will have to have hard copy output available.

The simplest way to obtain this is to log into the service once with the echo to printer enabled. This echoes all characters displayed to the line printer.

When writing a script file, lines that start with a semi colon (;) will be considered comments and not processed. Lines that contain only semi colons will serve a blank, or spacer, lines. You may have multiple statements on each line, separated by a colon (:).

The second method of creating a script file uses the powerful OnLine! "learn mode". This learn mode is a combination of two command keys, ALT-K and ATL-L. These two keys allow you to create script files in "real time", as you are actually online with the remote computer.

There is only one difference between these keys. This is the level of automation provided versus the amount of flexibility afforded. The ATL-L key is the one to use when you are buliding simple "prompt - response" sequences, and the ALT-K key allows you to enter the script command directly.

There is a quite detailed discussion of using the learn mode in the second manual section.

Please note that these keys will be ignored if there is a script file currently running. Suspend operation of the file first, and then you may use the learn mode.

Script file commands

The following table of commands details your possible choices when writing script files.

ABORT	Cancel operation of a script file.
ALARM [hz,ms]	Sound tone, with optional hertz and milliseconds.
ASK ["prompt"]	Prompt user for information. Used with IF.
BYE	Hangup telephone and terminate call.
CAPTURE option	Control capture buffer.
CLEAR i,p	Erase the current video contents, optional ink and paper.
COLOR i,p	Set display to specified ink and paper colors.
DO "filename"	Load and execute another script file.
IF	Logical operator. Tests ASK input or carrier.
JUMP label	Branch to label.
LABEL name	Marks line for branch with JUMP.
MESSAGE "string"	Display message on CRT.
NOT, !, -	Logical operator. Negation of condition in test.
OFFLINE	Identical to BYE command. Terminates call.
PRINTER switch	Toggle hard copy option on and off (printer).
REPLY	Send string of text out the modem.
RWIND	Re-start script from top. Usually on an error.
SBREAK [ms]	Send break, optional for specified milliseconds.
SKIP number	Bypass specified number of script lines.
WAIT condition	Delay until specific condition is met.
WHEN condition	Set up a background scan for a specific string.

At this time, we will address each of these commands in turn.

ABORT

The ABORT command will cause OnLine! to stop operation of a script file and return immediately to the terminal mode. This command is normally used when you have detected an error condition that is beyond possible correction, and wish to cease script file operation.

ALARM

This command is used when you wish to sound a tone to alert the operator to a particular condition has occurred, or that OnLine! has done something. When you specify the command, you may optionally pass it the desired tone in hertz and the desired duration in milliseconds. For example,

ALARM

will sound the default tone. By stringing these commands together, you obtain multi-tone sounds. For example, the two lines

ALARM 750,250
ALARM 500,250

will sound a high tone followed by a lower tone. Note the tone and duration parameters passed this time. The first value is the tone in hertz, and the second value is the duration in milliseconds.

ASK

The ASK command is used when you wish to prompt the operator for input. This can be used just to pause the script file until the operator signals to continue, or you may use it to

actually poll for information and act upon it. An example of this is,

ASK "Press a key to continue:"

which would just move to the next line when any key is pressed, and

**ASK "Check user mail?"
IF yY JUMP READMAIL
JUMP LOGOUT**

which polls for a single character answer to a question from the user and jumps to one of two locations depending upon it. Note please the case dependent nature of IF. (See also IF.) ASK may only be used to read in single character replies. This will be stored until the next ASK, so the IF does not have to immediately follow the ASK.

BYE

The BYE command is used when you wish OnLine! to terminate the telephone call and hangup the phone line. This will be very useful when creating completely automated download sessions, because without it, many packet networks such as TymeNet will not terminate the connection until a lengthy timeout has expired.

To use the command, enter

BYE

in the script file. At that point, OnLine! will immediately hangup the telephone and proceed to the next command in the script.

CAPTURE

This command gives you control over the capture buffer. It can accept parameters that control the capture buffer and captured data. With this command you may allocate and open, open, clear, close, load, or save the capture buffer. For example,

CAPTURE OPEN 256

will create a 256K capture buffer and open it. This will create an error if the requested buffer size exceeds the amount of free memory in the machine.

CAPTURE OPEN

will open an existing buffer. And,

CAPTURE CLEAR

will erase the data currently in the capture buffer. Finally,

CAPTURE CLOSE

will close the capture buffer.

You may also load data into the ASCII capture buffer. If there is no buffer space allocated, it will allocate the size of the file plus a few bytes for overhead. If the buffer space allocated is not enough to hold the file, the program will de-allocate the current buffer and attempt to re-allocate the buffer large enough to hold the file. If this also fails, the script file will abort with an error. For example,

CAPTURE LOAD "THISFILE.TXT"

will load the data from THISFILE.TXT into the buffer.

Also, you can store the current contents of the capture buffer into a disk file. For example,

CAPTURE SAVE "MYFILE.TXT"

would store all the data currently in the capture buffer into the disk file MYFILE.TXT. This would be used after you had captured some information under the control of a script file, and wished to save this data into a disk file. If the filename is not legal, an error will result.

CLEAR

The CLEAR command erases the current contents of the video display. This can be useful between services or when chaining script files. The simplest form of the command is

CLEAR

which simply clears the display with the current attributes for ink and paper color. Optionally, you may specify a new ink and paper color, as in

CLEAR 1,0

which would clear the display, setting it to normal white ink (1) on a normal blue paper (0).

COLOR

This command allows you to adjust the current display attributes with regards to ink and paper (foreground and background) colors. These new attributes will take effect at the next display access. For example, after

COLOR 0,2

the next text displayed will be normal blue ink on a normal black paper. By using the **COLOR** command, you may display error messages in special colors and then return to the "normal" display attributes.

Number	Color
0	Blue
1	White
2	Black
3	Orange

Note that these may be changed with Preferences.

DO

This command allows you to load and execute one script file from another. This can be very useful in decision making and implementing. For example,

```
ASK "Read the mail?"  
IF yY DO "READMAIL"  
DO "NOREAD"
```

will poll the user to see if they wish to read the mail. Based upon this answer, the script file runs one of two other script files, thereby removing the need for all of the code to fit into memory at one time.

With script file chaining, you can also operate script files right from the disk. And with more modules possible, the size of each does not need to be quite as great.

IF

The IF command allows you test the data read from an ASK and operate based upon a true-false condition. The ASK command will print a line of text on the screen and then poll the operator for input. (See ASK.) IF allows you to test that input, and act as instructed upon the outcome. The test string may be a single character or a list of characters. The test string is "case dependent", that is, all text will be tested exactly as it appears. To test for both an upper and lower case letter, you need to include both. For example,

```
ASK "Do you wish to continue?"  
IF yYcC JUMP KEEPGOING
```

will poll the operator for a character and then if the user types either an upper or lower case Y or C, it will continue. By using the NOT operator, you can reverse the emphasis of the test. For example,

```
ASK "Do you wish to continue?"  
IF NOT yYcC ABORT
```

is similar to the first command, but if the proper letter is NOT typed, it aborts. NOT may be abbreviated with an exclamation mark (!) or a hyphen (-).

You may also use IF to test for carrier. You may say, for example,

```
IF CARRIER JUMP NODIAL
```

which would tell the system to branch to the label NODIAL if carrier is present. CARRIER may also be expressed as ONLINE or abbreviated as a dollar sign (\$). It may also be combined with NOT to wait for carrier. For example,

```
LABEL CARRIERLOOP  
IF NOT CARRIER JUMP CARRIERLOOP
```

which will do just that. In its shortest form, this could be said as,

```
LABEL CARRIERLOOP  
IF !$ JUMP CARRIERLOOP
```

This test is most used in sounding ALARM tones to the operator when carrier is present to alert them to the fact that they are on-line.

JUMP

This command lets you branch to a preset location in the file. This is most often used with the IF and WHEN commands. For example,

```
ASK "Read the mail?"  
IF yY JUMP READMAIL  
ABORT  
LABEL READMAIL  
(commands to read the mail)
```

If the operator inputs a upper or lower case Y, it will branch around the ABORT and proceed with execution at the line signalled by the LABEL command.

Using it with the WHEN command is a bit harder. For a better explanation of WHEN see the WHEN sub-section. Suffice it to say that the command,

WHEN "CONNECT" JUMP STARTIT

will branch to the line labelled STARTIT any time the word CONNECT (all in caps) is received.

LABEL

The LABEL command allows you to "name" a line or location in the script file such that JUMP can locate it for a later branch. For example,

LABEL HERE

will bring the script file to that position every time a JUMP HERE command is used.

MESSAGE

The MESSAGE command allows you to send a message to the operator on the local screen without having anything sent to the host system. You may wish to preface the MESSAGE command with a CLEAR command if you would like to know that the screen is always empty. For example,

MESSAGE "Entering second phase of file."

will post that message to the screen. Preceding that command with a CLEAR command would cause that statement to get printed on a screen all by itself.

NOT

The NOT operator negates the test in an IF command. For details, see IF.

OFFLINE

The OFFLINE command is used when you wish OnLine! to terminate the telephone call and hangup the phone line. This will be very useful when creating completely automated download sessions, because without it, many packet networks such as TymeNet will not terminate the connection until a lengthy timeout has expired.

To use the command, enter

OFFLINE

in the script file. At that point, OnLine! will immediately hangup the telephone and proceed to the next command in the script. This command is identical to the BYE command, which was included for compatibility reasons.

PRINTER

This command will allow the user to turn on and off the echo to printer hardcopy option. This functions the same as the Printer Enable command from the main menu. For example,

PRINTER ON

will cause any characters received to be sent to both the video display and the line printer, and

PRINTER OFF

will shut off the printer output. This command will allow you to, under script file control, obtain hardcopy of various portions of your sessions without having the operator present.

REPLY

The REPLY command is used to send either a string of text or a macrokey out the modem. This command, combined with WAIT STRING, will make up your prompt - response auto-logon team. This command can be used ANY time that you wish to send a string of characters out the modem. For example,

REPLY "Bunch of characters"

will send that string of characters out the modem. Adding the vertical bar character to a string will cause a carriage return to be sent at that point. To send a macrokey, you would use

REPLY @F1

or

REPLY @1

and whatever string of characters is currently programmed onto macrokey #1 will be sent out the modem. In the case of command driven modems such as the Hayes Smartmodem, this can even be a modem command (i.e. a dial command). Note that macrokeys sent via the REPLY command will be affected by the keyboard translation table.

RWIND

The RWIND command tells OnLine! to start the script file over again from the top. This is usually implemented when an error condition is detected. You may use a JUMP to a label positioned at the top of the file, but this is both faster and does not use up the memory for storing the label.

Oftentimes, when you suspect that an error message might arise, using WHEN to look for the message and then doing an RWIND if it shows up makes an excellent error trapping routine.

SBREAK

The SBREAK command will send a true break signal, optionally for a specified number of milliseconds. The default length is 250 milliseconds. This command is the same as pressing ALT-C from the terminal mode. For example,

SBREAK

will send true break for 250 milliseconds, while

SBREAK 1000

will send true break for a full second.

SKIP

This command gives you the option of skipping over a specified number of lines in the script file. You may not skip past the end of file, nor may you skip "up" in a file by specifying a negative number. For example,

SKIP 3

will skip the next three lines of the script file. This command is useful in conjunction with conditional testing with the IF command in that it allows you to skip over undesired blocks of code, depending upon the outcome of the test. For example,

```
ASK "Do this next stuff?"  
IF lyY SKIP 3  
(3 lines of code to do stuff)  
(next line of code)  
will let you bypass code on command.
```

WAIT

The WAIT command allows you to tell OnLine! to wait until a specific condition has been met. You specify these conditions as an arguments to the WAIT command. Combined with the REPLY command, this is the “prompt” part of the “prompt – response” sequences. The conditions are:

ECHO	Wait for a carriage return from host
QUIET xx	Wait until the line is “quiet” (e.g. no characters received) for xx tenths of a second.
DELAY xx	Wait for xx tenths of a second.
CHAR “c”	Wait until character “c” is received.
FOR “c”	Same as CHAR.
STRING “string”	Wait until “string” is received.
UNTIL hh:mm	Wait until system clock equals “hh:mm”.
REPLY	Wait until current reply is sent.
MKEY	Wait until current macrokey is sent.
PROMPT xx	Wait until “xx” characters are received.

This command is a VERY powerful method of controlling the actions of a script file based upon input from the host system. For example,

```
WAIT STRING “Account ID:”  
REPLY “123ABC|”
```

tells OnLine! to wait until the host sends the string "Account ID:", and then respond with the string "123ABC" followed by a carriage return. You can see how simple this makes auto-logons. The learn mode, with the F9 key, builds a script file of WAIT STRINGs followed by REPLYs. All you do is edit the strings to be waited on and sent. The maximum length of a prompt string is 15 characters. More will cause an error.

When looking for only a single character, use WAIT CHAR. You can search for control characters with either one by using the "carat letter" style of designating control values. For example,

```
WAIT CHAR "^E"  
WAIT STRING "LOGIN:^E"
```

In the first, OnLine! simply waits for a CTL-E. In the second, OnLine! will wait for the specified string, but only that string FOLLOWED by a CTL-E will suffice. This also applies to WAIT FOR.

WAIT DELAY and WAIT QUIET are two powerful time-controlled factors. For example,

```
WAIT QUIET 20
```

tells OnLine! to wait until the line has been silent for 2 seconds, while

```
WAIT DELAY 20
```

tells OnLine! to wait 2 seconds no matter whether the line is silent or not.

WAIT UNTIL simply allows OnLine! to pause waiting for the system clock to reach a specified time before it continues. WAIT ECHO is used when you want OnLine! to wait until the

first carriage return is received from the host and then proceed with a task.

WAIT REPLY and **WAIT MKEY** perform essentially the same function, just in a slightly different manner. If you want to have multiple replies to a single prompt, this is possible. However, because one **REPLY** might not be completely finished transmitting before the next **REPLY** command is reached, it would cause the first **REPLY** to go unfinished.

To compensate for this, you may tell OnLine! to wait until the current **REPLY** (or macrokey, if you are replying with one of those) is completely sent before moving to the next script command. For example, if your file had

```
REPLY "First string|"  
REPLY "Second string|"
```

The first string would not be completely sent before OnLine!

reached the script command to start the next reply. You might end up with something that looked like

```
FirSecond string
```

instead of what you want. Therefore, you would modify the commands to

```
REPLY "First string|"  
WAIT REPLY  
REPLY "Second string|"
```

This will cause OnLine! to wait until the first reply is completely sent, and then it will proceed to the next

command, which starts the next reply. If you are using a macrokey instead, the commands would look like

```
REPLY @1  
WAIT MKEY  
REPLY @2
```

Different slightly, but performing the same function.

WAIT PROMPT lets you tell OnLine! how many characters to wait for without having to tell it exactly what they are. Let's take an example. Assume that we are connected to a service that, as it performs a specific task, prints the "*" character as each phase is complete. You wish to have OnLine! alert you when 6 of these characters have been received. First, you would more than likely do a WAIT STRING for the announcement that the operation was starting.

Example:

```
WAIT STRING "Starting computation"  
WAIT PROMPT 6  
ALARM
```

This would cause OnLine! to wait until the message "Starting computation" was received. And then, when the 6th character following that message was received (and remember this is ANY character, even carriage returns and line feeds or non-displayable control values), an alarm will be sounded.

By using the WAIT command, you can program OnLine! to respond to the output of a host system. Using REPLY to send data to the host, OnLine! can operate the remote system as if an operator was there. Using ASK and MESSAGE, it can

keep the operator posted as to its progress and make decisions based on input.

WHEN

The WHEN command is a special case that allows you to set up, in effect, an “interrupt condition”. What this means is that whenever a preset word or string is sent from the host, execution of the script file is interrupted and a statement(s) is executed. This helps to automate your sessions even more and makes an excellent error-trap.

For example,

```
WHEN “host system dropped” JUMP  
BLOWOUT
```

will cause OnLine! to branch to the portion of the script file named BLOWOUT whenever the message “host system dropped” is seen. It is also useful in repetitive situations. If a report will be sent paged and you wish to download it all, the command

```
WHEN “Press ENTER for more” REPLY |
```

will be useful. Each time the message is seen, a carriage return will be sent. This keeps the report going with one command.

You may only have one WHEN condition set at a time. Setting another will erase the first. To shut off the WHEN altogether, simply enter WHEN in the script file without a condition or command.

This is important. WHEN commands take a great deal of processor time. Each time a character is received, OnLine! must check for the entire WHEN condition. This can cause some “bogging down” and eventual character overrun. It is not

a good idea to leave a long WHEN condition in effect for a long time.

It is recommended that you set the WHEN at the time that you expect a potential problem and disable it as soon as the problem area is past or the problem is encountered.

NOTE: WHEN will no longer be effective if a script file terminates. So, whenever you DO set a when, install a loop beneath it to keep the program running in the script file.

Example:

```
WHEN "--More--" REPLY "|"
LABEL LOOP
IF CARRIER JUMP LOOP
```

This will cause the WHEN to continue being executed. If the script file indicator on the status line reads "Off", no WHEN is active.

General programming tips

Commands that prompt on strings, like WAIT STRING or WHEN, are very much at the mercy of line noise. Realize this, and use them cautiously. Keep the strings short, to minimize the chances of noise getting between characters. Error trap with a WHEN interrupt and a WAIT DELAY loop. Use the WAIT DELAY to wait while WHEN looks for the prompt. If WAIT DELAY has fallen through after a long period of time, noise probably got you, so shut off the WHEN and ship the REPLY anyway. WAIT PROMPT could also be useful.

Use the learn mode to build the prompt - response skeleton and then edit and optimize the script file. There is no

substitute for being on-line with a service when it comes to setting up the auto-logon.

Using the learn mode

When using the learn mode, you are actually building a script file. OnLine! can input the commands for you. You are building what we call a “prompt – response” style of script file. You will wait until a certain prompt is seen and then do something. Not as advanced as you can get, but effective and **VERY** easy to create! Your only choice is what you want to do when the prompt is seen. Do you want to:

- (1) Send a **REPLY**?
- (2) Use another script command?

When you get to the prompt that you wish to respond to, make this decision. Then, if you simply want to reply, key **ALT-L**. If you wish to enter another script file command, key **ALT-K** instead.

If you press this key in error, select **Cancel** and return to the terminal mode where you were.

In either case, OnLine! will display the last 15 characters received (which is your maximum prompt length), and place an edit cursor on the string. Using the edit keys, create the exact string you wish to respond to. You can replace lost characters, delete line noise, or just shorten it some (as per the tips section above). When the prompt string is as you wish, press **RETURN**. This will enter a **WAIT STRING** “prompt...” into the script file (where “prompt...” is whatever string you just created).

If you keyed **ALT-L**, you will be asked for the reply string. Enter it, placing a vertical bar (|) wherever you would press **RETURN**. When you are done, press **RETURN**. OnLine! will then send that to the host and store a **REPLY** “string...”

in the script file (where “string...” is the string you just created).

If you keyed ALT-K, you will be prompted for the entire script command. You may open the buffer, turn on the printer, sound an alarm, or anything else that you would normally do with a script file. If you keyed ALT-K and now decide that you simply wish to reply, then enter REPLY “string...”.

There is one important note that you should keep in mind when using the learn mode. If there is a script file active (the script indicator reads “On”), the ALT-K and ALT-L keys will not respond until you have suspended the script file

Changing the script files

It is always best to use a text editor or word processor to edit the files whenever possible. Files can be carriage return/line feed at the end of each line, just carriage returns, or just line feeds. OnLine! will not care.

When an unexpected prompt surfaces, and it will be a permanent prompt, make a note of it and edit the script file after you disconnect (with your text editor). Do not attempt online editing of script files. This will almost assuredly produce results OTHER than what you expect!

A

- Accessing information services, 1-14 to 1-15
- Amiga-keys, 1-16 to 1-19
- Apostrophe (') macrokey, 1-22, 3-5, 3-28
- Archive requester, i-3, 1-19, 1-21 to 1-22, 3-3 to 3-5
- Automating communications, 2-1 to 2-15. *See also* Script files

B

- Backup diskettes, 1-7 to 1-8
- Baud rate, 1-14 to 1-15, 1-24, 2-10, 3-17 to 3-18
- Binary files, 1-34
- Blank lines in files, 1-34
- Break key, 1-19
- Bulletin boards. *See* Information services

C

- Cable requirements, 1-6 to 1-7
- Capture buffer, 1-30 to 1-32
 - status messages
 - file, 2-12
 - free, 2-12
 - size, 2-11 to 2-12
 - status, 2-12
 - used, 2-12
- Capturing data
 - to disk, i-3, 1-31 to 1-32
 - to memory, 1-29 to 1-30

Capture menu, 1-30 to 1-32, 3-10 to 3-15. *See also*

Sending files

archive, 1-31, 3-11

C-delay, 3-14

clear, 1-31, 3-13

close, 3-13

display, 3-15

go, 3-14

L-delay, 3-14 to 3-15

open, 1-30, 3-11 to 3-13

printer, 3-15

prompt, 3-15

send, 1-33 to 1-34, 3-13 to 3-14

stop, 3-14

view, 1-30, 3-15

Carat (^) macrokey, 1-23, 3-5, 3-28

**Carriage return/line feed, 2-13 to 2-14,
3-22**

COM menu, 3-17 to 3-19

baud, 3-17 to 3-18

parity, 3-18

reset, 3-17

stops, 3-18

word, 3-18

Command

descriptions, 3-1 to 3-2

mode, 3-1

selecting, 3-1 to 3-2

Command menus

Capture menu, 3-10 to 3-15

archive, 3-11

C-delay, 3-14

clear, 3-13

Command menus**Capture menu (Continued)**

- close, 3-13
- display, 3-15
- go, 3-14
- L-delay, 3-14 to 3-15
- open, 3-11 to 3-13
- printer, 3-15
- prompt, 3-15
- send, 3-13 to 3-14
- stop, 3-14
- view, 3-15

COM menu, 3-17 to 3-19

- baud, 3-17 to 3-18
- parity, 3-18
- reset, 3-17
- stops, 3-18
- word, 3-18

Other menu, 3-19 to 3-24

- CR+LF, 3-22
- duplex, 3-21
- echo, 3-21
- flow, 3-23
- printer, 3-19
- terminal, 3-19 to 3-20
- width, 3-20
- window, 3-23 to 3-24

Phone menu, 3-24 to 3-27

- alternate, 3-25
- prefix, 3-26
- suffix, 3-27
- primary, 3-25
- retries, 3-25 to 3-26

Command menus (Continued)**Script menu, 3-15 to 3-17**

archive, 3-16

clear, 3-17

go, 3-16

resume, 3-16

stop, 3-16

view, 3-17

Service menu, 3-6 to 3-8

archive, 3-7

call, 3-7

hangup, 3-8

quit, 3-8

Tables menu, 3-27 to 3-31

display, 3-30

from-capture, 3-30

in-COM, 3-31

keyboard, 3-30

macrokey, 3-27 to 3-29

out-COM, 3-31

printer, 3-30

to capture, 3-31

Transfer menu, 3-8 to 3-10

protocol, 3-9

CRC-Xmodem, 3-10

HVP, 3-10

Xmodem, 3-10

receive, 3-9

send, 3-9

Communications parameters, 1-14 to 1-16

saving, 1-25 to 1-26

setting, 1-24 to 1-25

Configuration files. *See* Terminal files

Connecting with an information service,
 1-27 to 1-27
 errors, 1-28 to 1-29
Customization. *See* Terminal files

D

Dialing. *See* Connecting with an information service
Display_Beep() routine, i-3
Downloading data. *See* Capturing data
Duplex status, 2-14, 3-21

E

Echo on/off, 2-13, 3-21
Editing
 input line, 1-20
 script files, A-24
Errors in connections, 1-28 to 1-29

F

File transfer. *See* Sending files
Flow control, 3-23

H

Hardware requirements, 1-5 to 1-7

I

Information services, 1-8 to 1-9
 connecting with, 1-27 to 1-28
 log-on requirements, 1-14 to 1-15

K

Keyboard, 1-16 to 1-19

L

Learn mode, 2-1 to 2-3, 2-6 to 2-15, A-4, A-23 to A-24

Line feeds. *See* Carriage return/line feed

Line input requester, 1-19 to 1-20, 3-2 to 3-3

Logging on to information services, 1-14 to 1-15. *See also* Script files

M

Macrokey requester, 1-19, 1-22 to 1-23, 3-5 to 3-6
apostrophe ('), 1-22, 3-5
carat (^), 1-23, 3-5
reverse apostrophe ('), 1-22, 3-5
tilde (-), 1-22, 3-5
vertical bar (|), 1-22, 3-5

Macrokeys, 1-26 to 1-27, 3-27 to 3-29
apostrophe ('), 1-22, 3-5, 3-28
carat (^), 1-23, 3-5, 3-28
reverse apostrophe ('), 1-22, 3-5, 3-28
tilde (-), 1-22, 3-5, 3-28
vertical bar (|), 1-22, 3-5, 3-28

Main Menu, 1-17

Menus. *See* Command menus

Mouse usage, 1-11 to 1-12, 1-17

N

Null values, i-3

P

Parallel device, i-3

Parity check, 1-14 to 1-15, 1-24, 2-10 to 2-11, 3-18

Password requirements, 1-14 to 1-15

Percent sign (%) in strings, i-3

Phone menu, 3-24 to 3-27

alternate, 3-25

prefix, 3-26

carat (^), 3-27

reverse apostrophe ('), 3-26

tilde (~), 3-26

vertical bar (|), 3-27

suffix, 3-27

primary, 3-25

retries, 3-25 to 3-26

Phone numbers

primary, 2-10

retries, 2-10

setting, 1-23 to 1-24

Power-up default values, 1-15 to 1-16

Print

echo on/off, 2-12, 3-19

transcript of a session, 1-29

Protocol file transfers, 1-34 to 1-35

status, 2-13

Protocols, 3-9

CompuServe's B protocol, i-3

CRC-Xmodem, 3-10

HVP, 3-10

Xmodem, 3-10

R**Requesters**

- archive, 1-19, 1-21 to 1-22, 3-3 to 3-5
 - line input, 1-19 to 1-20, 3-2 to 3-3
 - macrokey, 1-19, 1-22 to 1-23, 3-5 to 3-6
- Reverse apostrophe (') macrokey, 1-22, 3-5, 3-28

S

Screen width setting, 3-20 to 3-21

Script commands, A-4 to A-22

- abort, A-6
- alarm, A-6
- ask, A-6 to A-7
- bye, A-7
- capture, A-8 to A-9
- clear, A-9
- color, A-9 to A-10
- do, A-10
- if, A-11 to A-12
- jump, A-12 to A-13
- message, A-13
- not, A-14
- offline, A-14
- printer, A-14 to A-15
- reply, A-15
- rwind, A-15
- sbreak, A-16
- skip, A-16
- wait, A-17 to A-21
- when, A-21 to A-22

Script menu, 3-15 to 3-17

- archive, 3-16
- clear, 3-17
- go, 3-16
- resume, 2-6, 3-16
- stop, 3-16
- suspend, 2-6
- view, 2-8, 3-17

Script files

- controlling, 2-5 to 2-6
- creating
 - with learn mode, 2-1 to 2-3, 2-6 to 2-15, A-4, A-23 to A-24
 - with text editor, 2-3, A-3 to A-4
- definition, 1-10, 2-3 to 2-4, A-1 to A-3
- editing, A-24
- examples, 2-4 to 2-5
- in the status window, 2-10
- programming tips, A-22

Sending files. *See also* Capture menu

- commands
 - C-delay, 1-33
 - go, 1-33
 - L-delay, 1-33
 - prompt, 1-33 to 1-34
 - stop, 1-33
- protocol file transfers, 1-34 to 1-35
- text buffer send, 1-32 to 1-33

Service menu, 3-6 to 3-8

- archive, 1-26, 3-7
- call, 3-7
- hangup, 3-8
- quit, 3-8

Starting Online!, 1-10 to 1-11

Status window, 2-9 to 2-15
 baud, 2-10
 CR+LF, 2-13 to 2-14
 duplex, 2-14
 echo, 2-13
 file, 2-12
 free, 2-12
 parity, 2-10
 primary, 2-10
 print, 2-12
 protocol, 2-13
 retries, 2-10
 script, 2-10
 service, 2-9 to 2-10
 size, 2-11 to 2-12
 status, 2-12
 stops, 2-11
 terminal, 2-13
 used, 2-12
 word, 2-11
Stop bits, 1-14 to 1-15, 1-24, 2-11
Strings, i-3
System requirements, 1-5 to 1-7

T

Tables menu, 3-27 to 3-31
 display, 3-30
 from-capture, 3-30
 in-COM, 3-31
 keyboard, 3-30
 macrokey, 3-27 to 3-29
 out-COM, 3-31
 printer, 3-30
 to capture, 3-31

Telephone dialing

- pulse, 1-28

- tone, 1-28

Telephone menu. *See* Phone menu**Terminal emulation, i-3, 2-13, 3-19 to 3-20****Terminal files**

- creating a new file, 1-14 to 1-16

- customizing, 1-26 to 1-27

 - macrokeys, 1-26 to 1-27

 - window size, 1-26

- definition, 1-12 to 1-13, A-1 to A-2

- loading an existing file, 1-13

Terminal mode

- definition, 1-11

Text buffer send, 1-32 to 1-34**Tilde (-) macrokey, 1-22, 3-5, 3-28****Transcripts**

- of a script file, 2-4 to 2-5

- of a session, 1-29

Transfer menu, 1-34 to 1-35, 3-8 to 3-10

- protocol, 3-9

 - CRC-Xmodem, 3-10

 - HVP, 3-10

 - Xmodem, 3-10

- receive, 3-9

- send, 3-9

Transferring files. *See* Sending files**Translation tables. *See also* Tables menu**

- editing, 3-29 to 3-30

V**Vertical bar (|) macrokey, 1-22, 3-5, 3-28****VT-100 emulation, i-3**

W

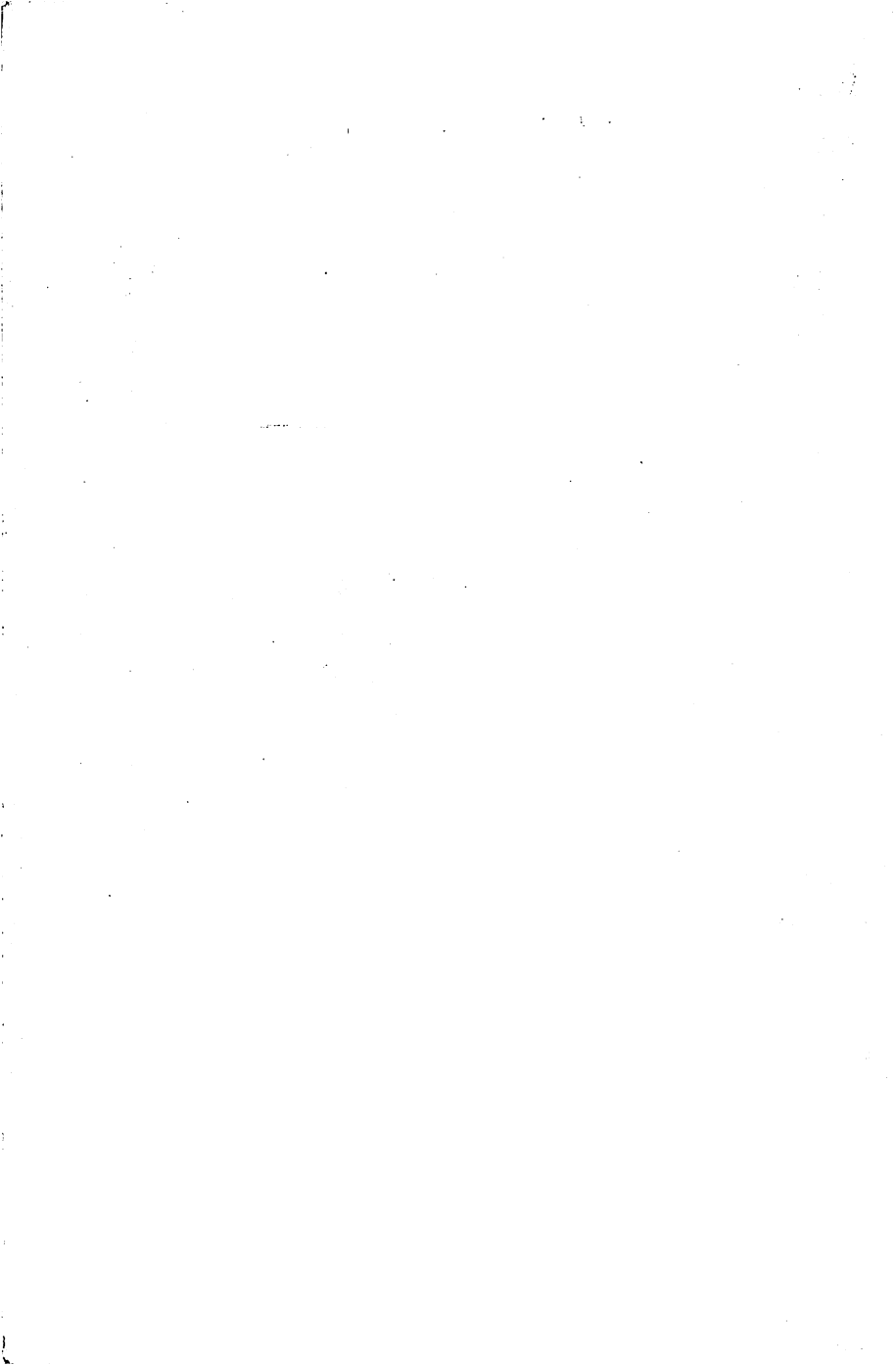
Width of screen, 3-20 to 3-21

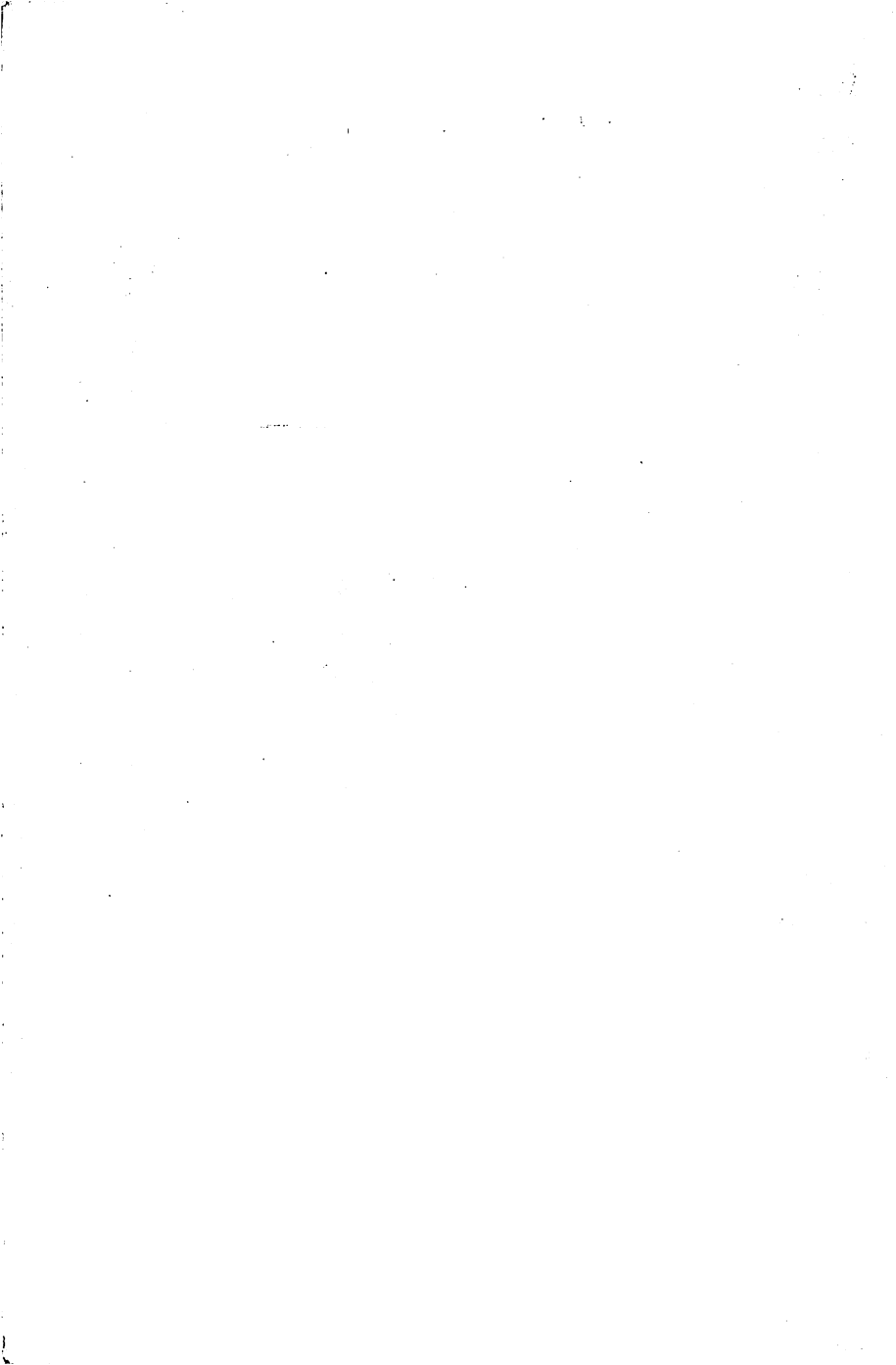
Window size, 1-26 to 1-27, 3-23 to 3-24

Word size, 1-14 to 1-15, 1-24, 2-11, 3-18

X

XMODEM protocol, 1-34





Telecommunications Plus with OnLine!

Micro-Systems Software proudly presents OnLine! for the Commodore Amiga personal computer. OnLine! combines features and convenience in a high quality package that will meet all your telecommunications needs.

OnLine! offers you the capability to use your Amiga as a window to the world of information that is just on the other side of your telephone. You can link up with commercial information services for stock quotes, airline information and reservations, technical databases, and thousands of other business and entertainment tasks. You can also plug into local bulletin board systems (BBS) and discover a new world of information and software for your computer.

Corporate users will discover that OnLine! is an excellent way to let your Amiga access data stored on the company's mainframe computer.

OnLine! is the finest program of its type available for the Commodore Amiga. You can't lose when you get "online" with OnLine!.

Check these exciting features:

- Simple pull-down menu interface.
- All program settings, including auto-logon scripts, may be stored in special terminal definition files for easy recall. You may store files for as many services as disk space permits.
- All file functions support DOS pathnames and hard disks.
- ASCII capture buffer for storing incoming data, with an optional overflow to a disk file.
- Captured data may be displayed or printed on the lineprinter.
- Data may be transmitted from the capture buffer to the host in either prompted mode (line by line), block mode (XON/XOFF control), or with user-defined delays between characters and/or lines.
- Support for 300 through 19200 baud. Any word length or parity supported.
- User-definable macrokeys can be used to transmit often used commands.
- Program can operate in full or half duplex.
- Auto-dial support for any command driven modem. Default setting supports Hayes and Hayes compatible modems.
- Allows two phone numbers for each service, with user-definable retries (up to 99). If connection is not made on the primary number, the alternate will be tried.
- 7 separate translation tables permit you to change any one byte value into any other value for all system devices.
- User can control window size (79x22 with frame, 80x23 without). ANSI escape sequences supported in TTY emulation mode.
- Special "chat" window provides local echo regardless of incoming data. Excellent for CompuServe's CB Simulator.
- Hardcopy support. OnLine! can send received characters to both the screen and the printer.
- OnLine! can use script files for automated operation (not just auto-logon, but entire automated sessions — even downloading).
- Script files can be created online, in a simple question and answer style, with OnLine!'s unique "learn mode".
- Direct to disk file transfer is supported in four different protocols: XMODEM (Christensen), XMODEM/CRC, Hayes Verification Protocol (compatible with SmartCom), and standard text capture to disk. XMODEM protocol is "relaxed" for use with services such as TymeNet.

OnLine!